Statistical Learning project

Nebbiai Giulio (2092296), Pittorino Francesco Pio (2090920)

Introduction

In the project a binary classification task is faced: the aim is indeed to define which are the medical and demographic variables that enhance the risk of having diabetes and to estimate a model that has an high True Positive Rate, useful for an early diagnosis. For our purpose we use diabetes dataset from The National Institute of Diabetes and Digestive and Kidney Diseases (NIDDK), one of the research institutes of the National Institutes of Health (NIH) in the United States.

Firstly we declare the libraries needed for the analysis:

```r
library(corrplot)
```

```
corrplot 0.92 loaded
```

```r
library(gridExtra)
```

```
Warning: package 'gridExtra' was built under R version 3.6.3
```

```r
library(grid)
library(ggplot2)
library(lattice)
library(correlation)
library(car)
```

```
Loading required package: carData
```

```r
library(MASS)
```

```
Warning: package 'MASS' was built under R version 3.6.3
```

```r
library(glmnet)
```

```
Warning: package 'glmnet' was built under R version 3.6.3

Loading required package: Matrix

Warning: package 'Matrix' was built under R version 3.6.3

Loaded glmnet 4.1-1
```

```r
library(class)
```

```
Warning: package 'class' was built under R version 3.6.3
```

```r
library(pROC)
```

```
Warning: package 'pROC' was built under R version 3.6.3

Type 'citation("pROC")' for a citation.


Attaching package: 'pROC'

The following objects are masked from 'package:stats':

    cov, smooth, var
```

Then, since in our analysis are present non-deterministic methods, we set the seed to make results replicable:

```
set.seed(123)
```

Data Presentation

Data Retrieval

The data set have been downloaded from Kaggle: link.

Kaggle is a platform for data sharing, that guarantees revision and validation of the set. The data-set has been loaded in RStudio environment through `read_csv()` function:

```
data = read.csv("C:\\Users\\nebbi\\OneDrive\\Desktop\\magistrale\\statistical 2\\archive\\
```

Data Description

The target population made up by females at least 21 years old having Pima Indian heritage. The data-set comprises of medical and demographic information as well with their diabetes status, indicating whether they are positive or negative for the disease. The aim of the study is indeed to identify the patterns in the data that lead patients to be affected by diabetes.

Data is made up by 768 examples described by 9 variables.

```
str(data)
```

```
'data.frame':   768 obs. of  9 variables:
 $ Pregnancies             : int  6 1 8 1 0 5 3 10 2 8 ...
 $ Glucose                 : int  148 85 183 89 137 116 78 115 197 125 ...
 $ BloodPressure           : int  72 66 64 66 40 74 50 0 70 96 ...
 $ SkinThickness           : int  35 29 0 23 35 0 32 0 45 0 ...
 $ Insulin                 : int  0 0 0 94 168 0 88 0 543 0 ...
 $ BMI                     : num  33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 0 ...
 $ DiabetesPedigreeFunction: num  0.627 0.351 0.672 0.167 2.288 ...
 $ Age                     : int  50 31 32 21 33 30 26 29 53 54 ...
 $ Outcome                 : int  1 0 1 0 1 0 1 0 1 1 ...
```

Here the first 6 observations:

```
head(data)
```

```
  Pregnancies Glucose BloodPressure SkinThickness Insulin  BMI
1           6     148            72            35       0 33.6
2           1      85            66            29       0 26.6
3           8     183            64             0       0 23.3
4           1      89            66            23      94 28.1
5           0     137            40            35     168 43.1
6           5     116            74             0       0 25.6
  DiabetesPedigreeFunction Age Outcome
1                    0.627  50       1
2                    0.351  31       0
3                    0.672  32       1
4                    0.167  21       0
5                    2.288  33       1
6                    0.201  30       0
```

Following is a brief description of each variable:

Pregnancies *Discrete variable*

It refers to the number of pregnancies of the individual.

Glucose *Continuous variable*

It measures the amount of glucose in the bloodstream at a given time.

BloodPressure Continuous *variable*

It measures the blood pressure at a given time.

Skin Thickness *Continuous variable*

It measures the skin thickness in mm. The values equal to zero are considered Not Available values.

Insulin *Continuous variable*

It measures the insulin level in the bloodstream.

BMI *Continuous variable*

It expresses the Body Mass Index of the individual, which is computed as:

$$BMI = \frac{\text{weight (kg)}}{\text{height (m)}^2}$$

Even in this case values 0 represent missing values.

DiabetesPedigreeFunction *Continuous variable*

It describes the occurrence of diabetes within the example's family, taking into account different generations. Higher values indicate how an high part of individual relatives were affected by diabetes.

Age *Continuous Variable*

It expresses the age of the individual.

Outcome *Binary variable*

It expresses if the individual is affected (1) or not (0) by diabetes. This is the response variable for the study.

Data Filtering

SkinThickness

Main considerations:

- NA values for feature SkinThickness represent about the 30% of the dataset.

```
sum(data$SkinThickness==0)/nrow(data)*100
```

```
[1] 29.55729
```

- Medical literature suggest that diabetes is correlated with skin dryness and sensitiveness to infection, but not with the thickness. We can further verify this anticipating the correlation analysis with the response variable:

```
cor(data$Outcome,data$SkinThickness)
```

```
[1] 0.07475223
```

- In medical applications is way more difficult measuring the skin thickness than detecting diabetes because it needs more sophisticated tools and more specialized healthcare professionals. Since the aim of the study is forecasting diabetes condition, this information can be neglected since too complex to obtain.

It's reasonable to just remove this feature from the data:

```
data=data[,-4]
ncol(data)
```

```
[1] 8
```

BloodPressure and *BMI*

The examples with missing values for BloodPressure predictor represent only the 4.5% of our data

```
sum(data$BloodPressure==0)/nrow(data)*100
```

```
[1] 4.557292
```

The examples with missing values for BMI feature represent only about the 1.5% of the dataset.

```
sum(data$BMI==0)/nrow(data)*100
```

```
[1] 1.432292
```

It's reasonable to just remove these rows from dataset since it would mean a little loss in terms of observations.

```
data=data[ data$BMI!=0 & data$BloodPressure!=0,]
```

Check of balanceness

Since the response variable in the data-set express the presence or not of an illness, the data it's likely to be unbalanced for this feature given its nature. We can verify that from the following table:

```
prop.table(table(data$Outcome))
```

```

        0         1 
0.6556927 0.3443073 
```

The imbalanceness is not that marked. It's not likely to represent a problem for modelling.

Exploratory Data Analysis

We perform an EDA on our data in order to better understanding the features distributions and if there are meaningful patterns or relationship between variables. In order to develop more understandable results, we grouped the features into two clusters given the semantic of each variable:

- Demographic features: *Pregnancies*, *BMI* and *Age*
- Medical condition features: *Glucose*, *BloodPressure*, *Insuline*, *DiabetesPedigreeFunction*

Summary Statistics

Firstly, thanks to `summary()` function we explore the descriptive statistics over all features in our data.

```
print("PREGNANCY:")
```

```
[1] "PREGNANCY:"
```

```
print(summary(data$Pregnancies))
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  0.000   1.000   3.000   3.859   6.000  17.000 
```

```
print("GLUCOSE:")
```

```
[1] "GLUCOSE:"
```

```
print(summary(data$Glucose))
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
      0      99     117     121     141     199 
```

```
print("BLOOD PRESSURE")
```

```
[1] "BLOOD PRESSURE"
```

```r
print(summary(data$BloodPressure))
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  24.00   64.00   72.00   72.37   80.00  122.00 
```

```r
print("INSULIN")
```

```
[1] "INSULIN"
```

```r
print(summary(data$Insulin))
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
   0.00    0.00   46.00   83.95  130.00  846.00 
```

```r
print("BMI")
```

```
[1] "BMI"
```

```r
print(summary(data$BMI))
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  18.20   27.50   32.40   32.47   36.60   67.10 
```

```r
print("DIABETES PEDIGREE FUNCTION:")
```

```
[1] "DIABETES PEDIGREE FUNCTION:"
```

```r
print(summary(data$DiabetesPedigreeFunction))
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 0.0780  0.2450  0.3780  0.4741  0.6270  2.4200 
```

```
print("AGE:")
```

```
[1] "AGE:"
```

```
print(summary(data$Age))
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  21.00   24.00   29.00   33.32   41.00   81.00 
```

```
print("OUTCOME:")
```

```
[1] "OUTCOME:"
```

```
print(summary(data$Outcome))
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 0.0000  0.0000  0.0000  0.3443  1.0000  1.0000 
```

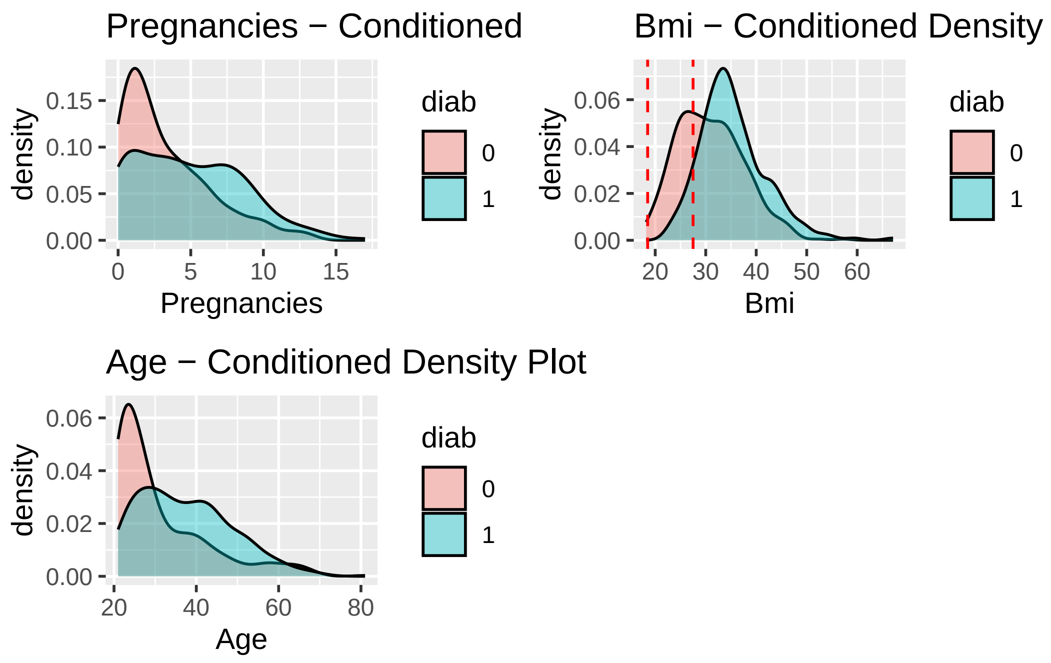
Conditioned Distributions

We now explore through graphs the conditioned distributions of our predictors.

```
diab=as.factor(data$Outcome)
#Pregnancies
a = ggplot(data, aes(x = data$Pregnancies, fill = diab)) +
  geom_density(alpha = 0.4) +
  ggtitle("Pregnancies - Conditioned Density Plot") +xlab("Pregnancies")
# Bmi
#bmi_classes outlines three different classes:
#<18.5 -> underweight
#from 18.5 to 27.5 normal weight
#>27.5 overweight
bmi_classes <- c(18.5, 27.5)
b = ggplot(data, aes(x = data$BMI, fill = diab)) +
  geom_density(alpha = 0.4) +geom_vline(xintercept =
                                          bmi_classes,
                                               linetype = "dashed",
                                               color = "red")+
  ggtitle("Bmi - Conditioned Density Plot") +xlab("Bmi")
# Glucose
c = ggplot(data, aes(x = data$Glucose, fill = diab)) +
  geom_density(alpha = 0.4) +
  ggtitle("Glucose level -Conditioned Density Plot") +xlab("Glucose")
#Blood_pressure
d = ggplot(data, aes(x = data$BloodPressure, fill = diab)) +
  geom_density(alpha = 0.4) +
  ggtitle("Blood Pressure Level - Conditioned Density Plot") +xlab("BloodPressure")
#Insulin
e = ggplot(data, aes(x = data$Insulin, fill = diab)) +
  geom_density(alpha = 0.4) +
  ggtitle("Insulin - Conditioned Density Plot") +xlab("Insuline")
#DiabetesPedigreeFunction
f = ggplot(data, aes(x = data$DiabetesPedigreeFunction, fill = diab)) +
  geom_density(alpha = 0.4) +
  ggtitle("Diabetes Pedigree Function - Conditioned Density Plot")+xlab("DiabetesPedigreeF
#Age
g = ggplot(data, aes(x = data$Age, fill = diab)) +
  geom_density(alpha = 0.4) +
  ggtitle("Age - Conditioned Density Plot") +xlab("Age")
```

Demographic Features

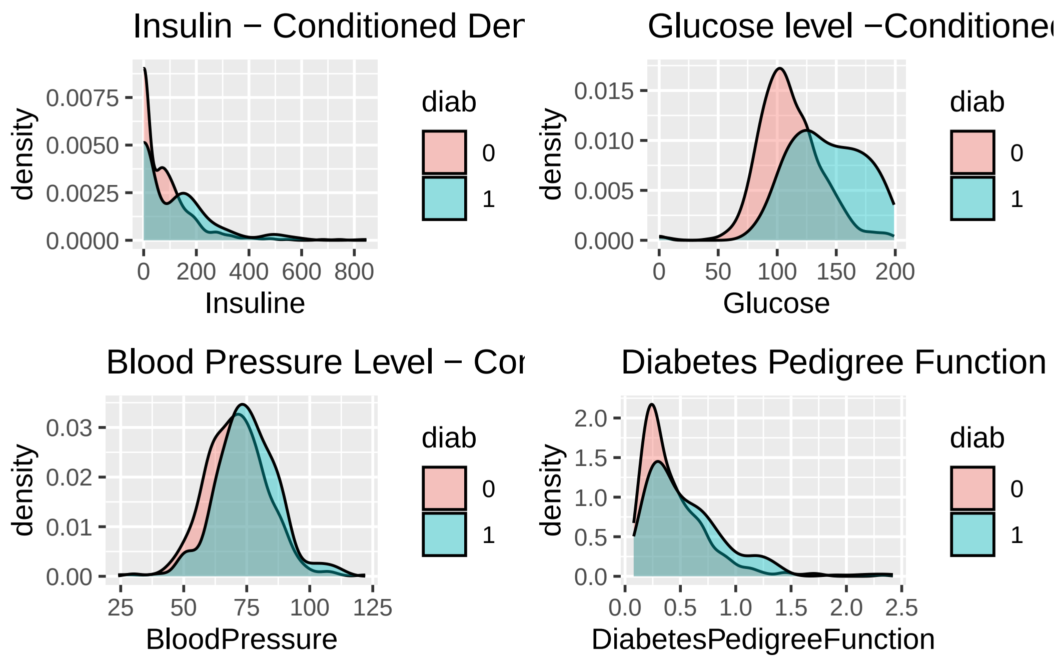
```
grid.arrange(a, b, g, nrow = 2)
```



About this first features cluster, the conditioned density plots show how the majority of people affected by diabetes in our data-set is old and have had a considerable number of pregnancies, but the diabetes conditioned distributions have more variance . For what concerns BMI feature, the diabetes affected population is made up by mostly obese women and both distribution have similar variance.

Medical Condition Features

```
grid.arrange(e,c,d,f,  nrow = 2)
```



For the second cluster of features instead we can identify two subset:

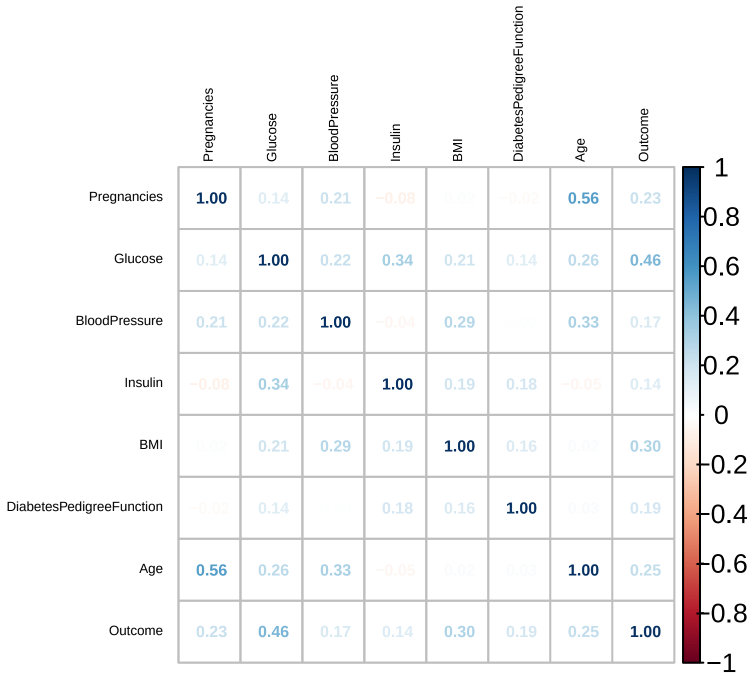
- The first is composed by *BloodPressure* and *DiabetesPedigreeFunction* that don't present really different density plots conditioning on the response variable.
- The second is composed by *Glucose* and *Insulin*, in which the distribution related to examples with diabetes has a higher campionary mean.

Correlation Analysis

The graphical correlation matrix below displays with more vivid colors higher correlations: in particular in blue the positive ones and in red the negative ones. Correlation between variables express the linear association between them and measure how variations in a variable are related to the variations of the other. From a first sight it's clear how all variables are not correlated or represent risk factors since positively correlated with diabetes: there are no protecting factors in the data . We can notice a more intense positive correlation between variable *Outcome* and variable *Glucose* (equal to 0.46), and between variable *Outcome* and variable *BMI* (equal to 0.30). Between risk factors the higher correlation are between *Glucose* and *Insulin.* This is reasonable since as we digest carbohydrates, pancreas recognizes higher level of glucose in the bloodstream and releases insulin. The matrix doesn't highlight other important correlations between risk factors, except for *Age* and *Pregnancies*, which is logical.

```r
correlation_matrix=cor(data)
```

```r
corrplot(correlation_matrix,
method ="number",
diag = TRUE,
tl.cex = 0.4,
number.cex = 0.5,
tl.col = "black")
```



We further analyze partial correlations, that also measure the linear association between two variables, but controlling the effects of other features keeping them fixed.

```r
print(correlation(data,partial=TRUE))
```

```
# Correlation Matrix (pearson-method)

Parameter1                |                Parameter2 |         r |          95% CI | t(727) |
--------------------------------------------------------------------------------------------
Pregnancies               |                   Glucose |     -0.04 | [-0.12,  0.03] |  -1.15 | >
Pregnancies               |             BloodPressure |      0.03 | [-0.04,  0.11] |   0.91 | >
Pregnancies               |                   Insulin |     -0.05 | [-0.12,  0.02] |  -1.32 | >
Pregnancies              |                      BMI |     -0.02 | [-0.10,  0.05] |  -0.64 | >
Pregnancies              | DiabetesPedigreeFunction |     -0.05 | [-0.12,  0.02] |  -1.41 | >
Pregnancies              |                      Age |      0.50 | [ 0.45,  0.55] |  15.65 | <
Pregnancies              |                  Outcome |      0.13 | [ 0.06,  0.21] |   3.67 | 0
Glucose                  |            BloodPressure |      0.14 | [ 0.07,  0.21] |   3.79 | 0
Glucose                  |                  Insulin |      0.33 | [ 0.26,  0.39] |   9.30 | <
Glucose                  |                      BMI | 2.57e-03 | [-0.07,  0.08] |   0.07 | >
Glucose                  | DiabetesPedigreeFunction |      0.01 | [-0.06,  0.08] |   0.31 | >
Glucose                  |                      Age |      0.16 | [ 0.09,  0.23] |   4.33 | <
Glucose                  |                  Outcome |      0.37 | [ 0.31,  0.43] |  10.74 | <
BloodPressure            |                  Insulin |     -0.13 | [-0.20, -0.05] |  -3.41 | 0
BloodPressure            |                      BMI |      0.29 | [ 0.23,  0.36] |   8.29 | <
BloodPressure            | DiabetesPedigreeFunction |     -0.04 | [-0.12,  0.03] |  -1.19 | >
BloodPressure            |                      Age |      0.23 | [ 0.16,  0.30] |   6.42 | <
BloodPressure            |                  Outcome |     -0.04 | [-0.11,  0.04] |  -0.98 | >
Insulin                  |                      BMI |      0.15 | [ 0.07,  0.22] |   3.96 | 0
Insulin                  | DiabetesPedigreeFunction |      0.13 | [ 0.05,  0.20] |   3.43 | 0
Insulin                  |                      Age |     -0.06 | [-0.13,  0.02] |  -1.56 | >
Insulin                  |                  Outcome |     -0.04 | [-0.11,  0.04] |  -0.96 | >
BMI                      | DiabetesPedigreeFunction |      0.09 | [ 0.02,  0.16] |   2.48 | 0
BMI                      |                      Age |     -0.10 | [-0.17, -0.03] |  -2.74 | 0
BMI                      |                  Outcome |      0.23 | [ 0.16,  0.30] |   6.50 | <
DiabetesPedigreeFunction |                      Age |      0.03 | [-0.04,  0.11] |   0.89 | >
DiabetesPedigreeFunction |                  Outcome |      0.12 | [ 0.05,  0.19] |   3.37 | 0
Age                      |                  Outcome |      0.07 | [ 0.00,  0.14] |   1.89 | 0

p-value adjustment method: Holm (1979)
Observations: 729
```

The partial correlations show that, in our dataset, only one partial correlation describes a strong association between two variables, after controlling for the effect of other variables in the set: the one between Age and Pregnancies. The other values of the partial correlations don't reveal the presence of other strong associations; indeed, many of these correlations are not significant.

Data-set splitting

It's necessary now to split data into:

- training set, that will be used to train the models;
- test set, used to evaluate the performance of these models.

Training set is made up by 75% of the original dataset.

```r
#Randomly choose a set of indices to split data set.seed(123)
random_indx = sample(1:nrow(data),
                     size = round(0.75 * nrow(data)),replace = FALSE)
#Creation of training set:
train = data[random_indx, ]
#Creation of test set:
test = data[-random_indx, ]
```

Thanks to this split we can evaluate the models' performance over some unseen data.

Logistic regression models

We define some logistic regression models to predict the probability to have diabetes, in order to classify whether a patient is diabetic or not. The logistic regression model is used to determine the probability of a subject to belong to class 1 of the binary response variable, based on its independent variables values. It's based on the logit, the function defined as the logarithm of the ratio between, in our case, the probability of having diabetes and the probability to don't have it. We define different models:

- The first defined using all features : the complete logistic model. It's moreover performed a VIF test to asses multicollinearity between the variables
- The second defined using Backward elimination, neglecting multicollinear variables (if there are any)
- The third defined using Forward selection, neglecting multicollinear variables (if there are any).

Complete Logistic Regression model

```r
model_1 = glm(data = train, train$Outcome ~ .,family = "binomial")
s_1=summary(model_1)
s_1
```

```
Call:
glm(formula = train$Outcome ~ ., family = "binomial", data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.4800  -0.7352  -0.4113   0.7537   2.7766

Coefficients:
                           Estimate Std. Error z value Pr(>|z|)
(Intercept)              -8.5555088  0.9139314  -9.361  < 2e-16 ***
Pregnancies               0.1361960  0.0385749   3.531 0.000414 ***
Glucose                   0.0324979  0.0042235   7.695 1.42e-14 ***
BloodPressure            -0.0121411  0.0096852  -1.254 0.209997
Insulin                  -0.0008140  0.0009525  -0.855 0.392765
BMI                       0.1066253  0.0184344   5.784 7.29e-09 ***
DiabetesPedigreeFunction  0.7988590  0.3442650   2.320 0.020315 *
Age                       0.0098462  0.0108378   0.909 0.363611
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 711.41  on 546  degrees of freedom
Residual deviance: 516.51  on 539  degrees of freedom
AIC: 532.51

Number of Fisher Scoring iterations: 5
```

```r
p_value= 1 - pchisq(s_1$null.deviance-s_1$deviance, 7)
p_value
```

```
[1] 0
```

In this model, most of the covariates result highly significant, except for "*Insulin*" ,"*Age*" and "*BloodPressure*". Moreover as we could expect from the correlation analysis each significant coefficient brings an increase in the logit of having diabetes related to the probability of the response variable.

From Residual deviance value, we can see that the logistic model is able to explain a significant portion of the Residual deviance compared with the null model.

To assess the presence of multicollinearity among the independent variables, it's introduced the *VIF* (*Variance Inflation Factor)* for all the covariates, that indicates if one independent variable is more related to another independent variable than to the response variable. One variable is affected by multicollinearity if its VIF value is larger than 1/(1-R2) (R2 is the Multiple R-squared of the regression).

```
# Compute the VIF
vif(model_1)
```

```
             Pregnancies                  Glucose            BloodPressure
                1.397697                 1.256495                 1.265047
                 Insulin                      BMI DiabetesPedigreeFunction
                1.246988                 1.155411                 1.020217
                     Age
                1.518657
```

```
#Compute the threshold
r2_1<- 1 - (s_1$deviance/s_1$null.deviance)
1/(1-r2_1)
```

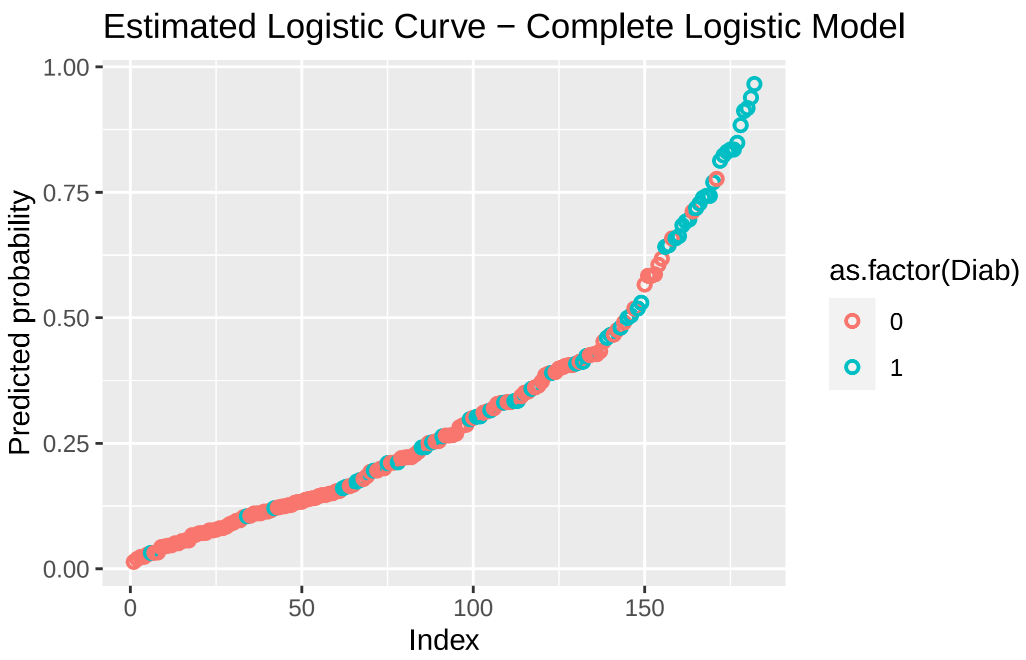
```
[1] 1.377341
```

As it's showed by the results, this model is affected by multicollinearity: VIF values of "*Pregnancies*" and "*Age*" are over the threshold value of 1,35. We could expect the result from correlation analysis. We decided to delete "*Age*" because of its highest VIF value and also because it doesn't turn out to be significant.

We can therefore proceed with the predictions. As the aim of the study is predicting diabetes, we'd like to have a model that doesn't misclassify an example if labelled as affected by the illness. To do so we set the threshold to 0.3 instead of the classic 0.5.

```
#Prediction
pred_model_1<- predict(model_1, test, type = "response")
#Conversion of the probabilities through the threshold of 0.3
pred_model_1r<- ifelse(pred_model_1 > 0.3, 1, 0)
```

```
predicted_data= data.frame(prob.of.Diab =
                             pred_model_1, Diab = test$Outcome)
predicted_data= predicted_data[order(predicted_data$prob.of.Diab,
                                     decreasing = FALSE),]
predicted_data$rank= 1:nrow(predicted_data)
ggplot(data = predicted_data, aes(x =
                                    rank, y = prob.of.Diab)) +
geom_point(aes(color = as.factor(Diab)),
           alpha = 1, shape = 1, stroke = 1) +
xlab("Index")+
ylab("Predicted probability")+
ggtitle("Estimated Logistic Curve - Complete Logistic Model")
```



We further calculate the confusion matrix in order to compute some evaluation metrics, in particular Accuracy and Sensitivity.

Accuracy measures the of prediction on overall data:

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{True Positives} + \text{True Negatives} + \text{False Positives} + \text{False Negatives}}$$

Sensitivity, also referred as True Positive Rate, is a measure the ability of a model to detect positive instances correctly.

$$\text{Sensitivity} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

```
#Confusion matrix
c_1=table(test$Outcome, pred_model_1r)
c_1
```

```
   pred_model_1r
     0  1
  0 84 41
  1 15 42
```

Accuracy:

```
accuracy_1 = (c_1[1,1]+c_1[2,2])/nrow(test)

accuracy_1
```

```
[1] 0.6923077
```

Sensitivity:

```
sensitivity_1 = c_1[2,2]/(c_1[2,2]+c_1[2,1])

sensitivity_1
```

```
[1] 0.7368421
```

Logistic with Stepsize Selection

The optimal collection of predictor variables for our logistic regression model are chosen using the variable selection method known as Stepwise Selection. It's a process that begins with an initial model and gradually adds or subtracts predicting variables based on, in our case, the Akaike Information Criteria (AIC), which is a measure that combines the model's complexity (expressed in the number of parameters) and data fit (expressed in the form of two times the loglikelihood)

$$AIC = 2 * k - 2 * log(L)$$

where: k= # of parameters and L=maximum value of likelihood function computed on the model. The lower the AIC, the better is the model. Based on choice of the initial model, there are two distinct procedures:

- Forward selection: we start with the Null model
- Backwards elimination: we start with the Complete model.

Given the previous result for VIF test we decided to apply both procedure to the model estimated on all predictors except *Age*.

Backwards elimination

```r
model_opt <- glm(data = train, Outcome ~  Pregnancies +
                   Glucose + BloodPressure + BMI +
                   DiabetesPedigreeFunction +
                   Insulin, family = binomial())
model_step_back <- stepAIC(model_opt, direction = "back" ,
trace = FALSE)
```

```r
s_back=summary(model_step_back)
s_back
```

```
Call:
glm(formula = Outcome ~ Pregnancies + Glucose + BMI + DiabetesPedigreeFunction,
    family = binomial(), data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.6252  -0.7228  -0.4166   0.7564   2.7498

Coefficients:
                          Estimate Std. Error z value Pr(>|z|)
(Intercept)              -8.779971   0.801322 -10.957  < 2e-16 ***
Pregnancies               0.146386   0.033051   4.429 9.46e-06 ***
Glucose                   0.031379   0.003752   8.364  < 2e-16 ***
BMI                       0.098020   0.017443   5.619 1.92e-08 ***
DiabetesPedigreeFunction  0.785279   0.339800   2.311   0.0208 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 711.41  on 546  degrees of freedom
Residual deviance: 519.01  on 542  degrees of freedom
AIC: 529.01

Number of Fisher Scoring iterations: 5
```

```
p_value= 1 - pchisq(s_back$null.deviance-s_back$deviance, 4)
p_value
```

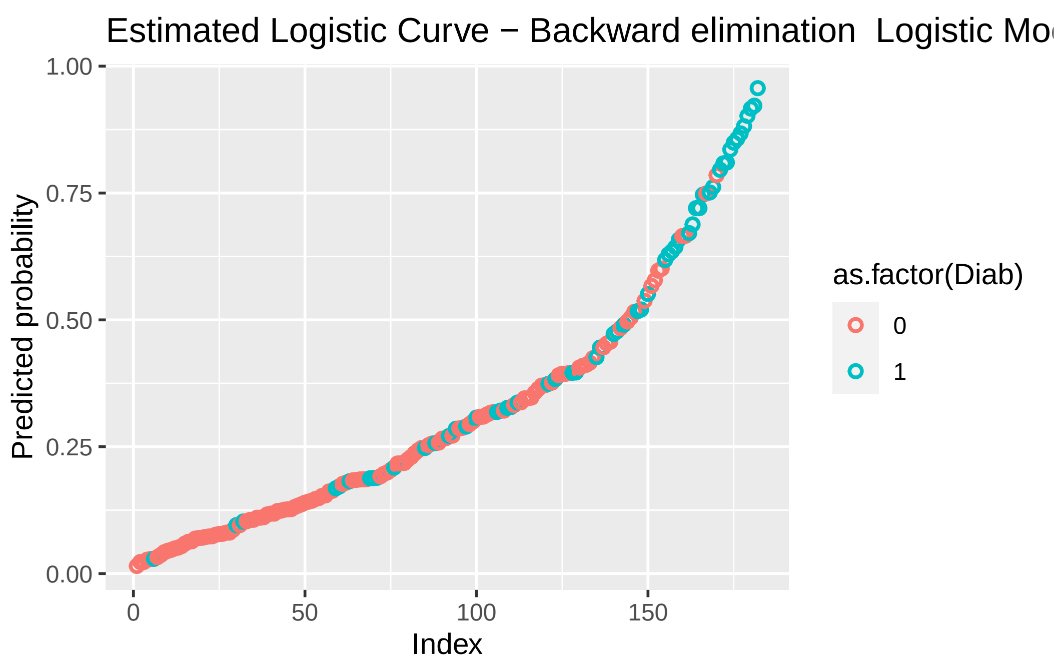
```
[1] 0
```

The estimated model doesn't comprehend coefficient related to *Insulin, and BloodPressure*; which were not significant in the complete model.

We can therefore proceed with the predictions. The threshold is set as before:

```
#Prediction
pred_model_back<- predict(model_step_back, test, type = "response")
#Conversion of the probabilities through the threshold of 0.3
pred_model_backr<- ifelse(pred_model_back > 0.3, 1, 0)
```

```
predicted_data= data.frame(prob.of.Diab = pred_model_back,
                           Diab = test$Outcome)
predicted_data= predicted_data[order(predicted_data$prob.of.Diab,
                                     decreasing = FALSE),]
predicted_data$rank= 1:nrow(predicted_data)
ggplot(data = predicted_data, aes(x = rank,
                                  y = prob.of.Diab)) +
geom_point(aes(color = as.factor(Diab)),
           alpha = 1, shape = 1, stroke = 1) +
xlab("Index")+
ylab("Predicted probability")+
ggtitle("Estimated Logistic Curve - Backward elimination  Logistic Model")
```



We compute the confusion matrix and the previous metrics:

```
#Confusion matrix
c_back=table(test$Outcome, pred_model_backr)
c_back
```

```
  pred_model_backr
     0  1
 0  84 41
 1  15 42
```

Accuracy:

```
accuracy_back = (c_back[1,1]+c_back[2,2])/nrow(test)

accuracy_back
```

```
[1] 0.6923077
```

Sensitivity:

```r
sensitivity_back = c_back[2,2]/(c_back[2,2]+c_back[2,1])

sensitivity_back
```

```
[1] 0.7368421
```

Forward selection

```r
model_step_for <- stepAIC(model_opt, direction = "forward" ,

trace = FALSE)
```

```r
s_for=summary(model_step_for)
s_for
```

```

Call:
glm(formula = Outcome ~ Pregnancies + Glucose + BloodPressure +
    BMI + DiabetesPedigreeFunction + Insulin, family = binomial(),
    data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.4984  -0.7441  -0.4123   0.7517   2.8168

Coefficients:
                           Estimate Std. Error z value Pr(>|z|)
(Intercept)              -8.4547982  0.9060975  -9.331  < 2e-16 ***
Pregnancies               0.1523735  0.0343729   4.433 9.30e-06 ***
Glucose                   0.0333498  0.0041405   8.055 7.98e-16 ***
BloodPressure            -0.0103617  0.0094848  -1.092   0.2746
BMI                       0.1047473  0.0182779   5.731 9.99e-09 ***
DiabetesPedigreeFunction  0.8070685  0.3437522   2.348   0.0189 *
Insulin                  -0.0008699  0.0009501  -0.916   0.3599
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)
```

```
    Null deviance: 711.41  on 546  degrees of freedom
Residual deviance: 517.33  on 540  degrees of freedom
AIC: 531.33

Number of Fisher Scoring iterations: 5
```

```
p_value= 1 - pchisq(s_for$null.deviance-s_for$deviance, 6)
p_value
```

```
[1] 0
```

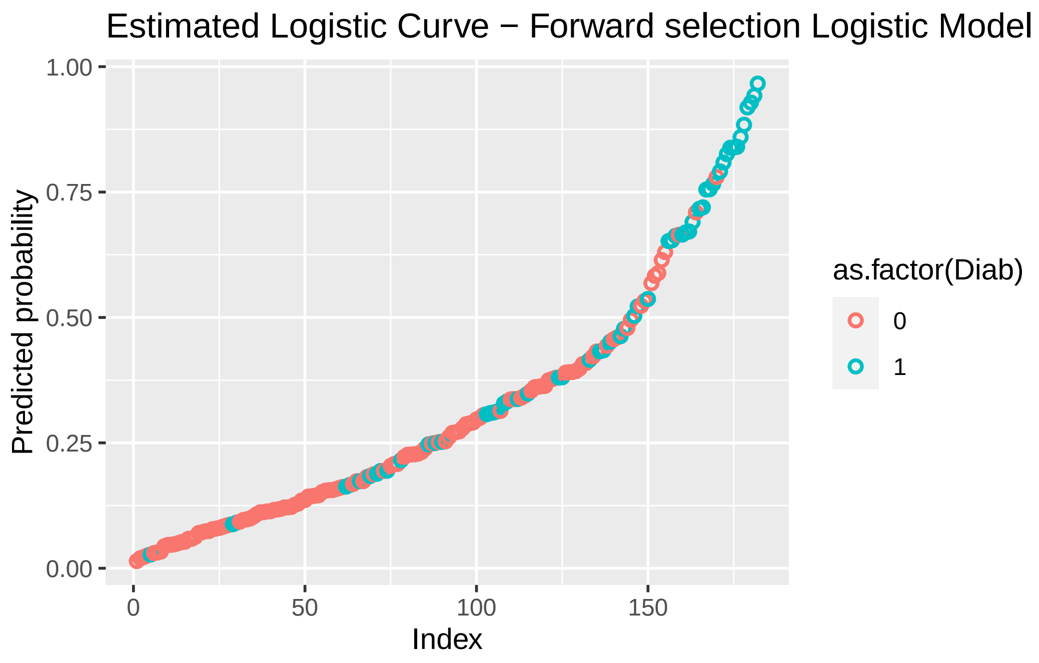
In the resulting model are included all covariates taken into account by the model that does not contain "Age" as it is affected by multicollinearity. "*Insulin*" and "*BloodPressure*" are still the only variables that are not statistically significant .

It is clear from the difference between the two deviances that the model is well fitted on the data.

We can therefore proceed with the predictions:

```
#Prediction
pred_model_for<- predict(model_step_for, test, type = "response")
#Conversion of the probabilities through the threshold of 0.3
pred_model_forr<- ifelse(pred_model_for > 0.3, 1, 0)
```

```
predicted_data= data.frame(prob.of.Diab = pred_model_for,
                           Diab = test$Outcome)
predicted_data= predicted_data[order(predicted_data$prob.of.Diab,
                                     decreasing = FALSE),]
predicted_data$rank= 1:nrow(predicted_data)
ggplot(data = predicted_data, aes(x = rank, y = prob.of.Diab)) +
          geom_point(aes(color = as.factor(Diab)),
          alpha = 1, shape = 1, stroke = 1) +
xlab("Index")+
ylab("Predicted probability")+
ggtitle("Estimated Logistic Curve - Forward selection Logistic Model")
```



```
#Confusion matrix
c_for=table(test$Outcome, pred_model_forr)
c_for
```

```
   pred_model_forr
     0  1
  0 87 38
  1 14 43
```

Accuracy:

```
accuracy_for = (c_for[1,1]+c_for[2,2])/nrow(test)

accuracy_for
```

```
[1] 0.7142857
```

Sensitivity:

```
sensitivity_for = c_for[2,2]/(c_for[2,2]+c_for[2,1])

sensitivity_for
```

```
[1] 0.754386
```

Shrinkage methods

Another way to select the best independent variables that allows forecasting the diabetes status is represented by shrinkage methods. Moreover they are useful to avoid Overfitting. In these algorithms, to compute the coefficients of the regression, we minimize a function made up by squared errors and a penalty term that makes coefficients estimates shrinking towards 0. In particular we are fitting:

- Ridge Regression model, that uses quadratic shrinkage : in the loss we consider the sum of squared coefficients multiplied by the tuning parameter lambda.
- Lasso Regression model, that uses absolute value shrinkage: in the loss we consider the sum of absolute value coefficients multiplied by the tuning parameter lambda.

The tuning parameter plays an important role in shrinkage methods, since higher the value, bigger will be the penalization. In both methods we use cross validation to choose the lambda which minimizes the misclassification error (maximizes the accuracy). It's important to note that, for their different loss formulation, Lasso regression is able to shrink parameter to exactly zero (and so it performs an automatic predictors selection), while Ridge is not capable of this.

Ridge Regression

```
#In order to use glmnet library for both Ridge and
#Lasso regression we have to create both a train
#and a test set that don't contain our response
#variable and convert it into matrix object

train_for_shrinkage=as.matrix(train[,1:7])
test_for_shrinkage=as.matrix(test[,1:7])
```

```
model_ridge=cv.glmnet(train_for_shrinkage, train$Outcome,

alpha = 0, family = "binomial", type.measure = "class")
```

```
model_ridge
```

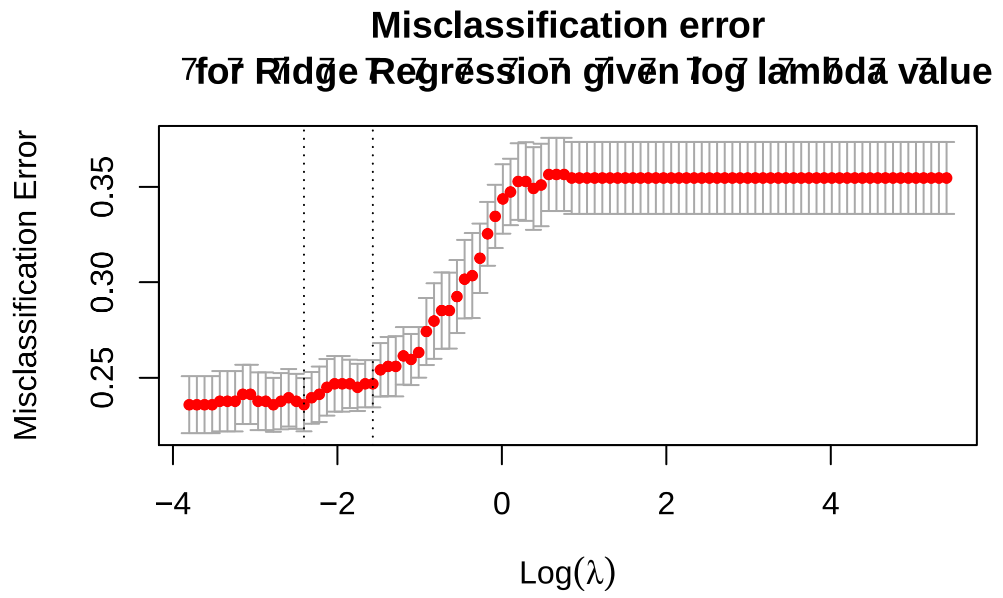
```
Call:  cv.glmnet(x = train_for_shrinkage, y = train$Outcome, type.measure = "class",      al

Measure: Misclassification Error

     Lambda Index Measure      SE Nonzero
min 0.09012    85  0.2358 0.01397       7
1se 0.20818    76  0.2468 0.01237       7
```

```
coef(model_ridge)
```

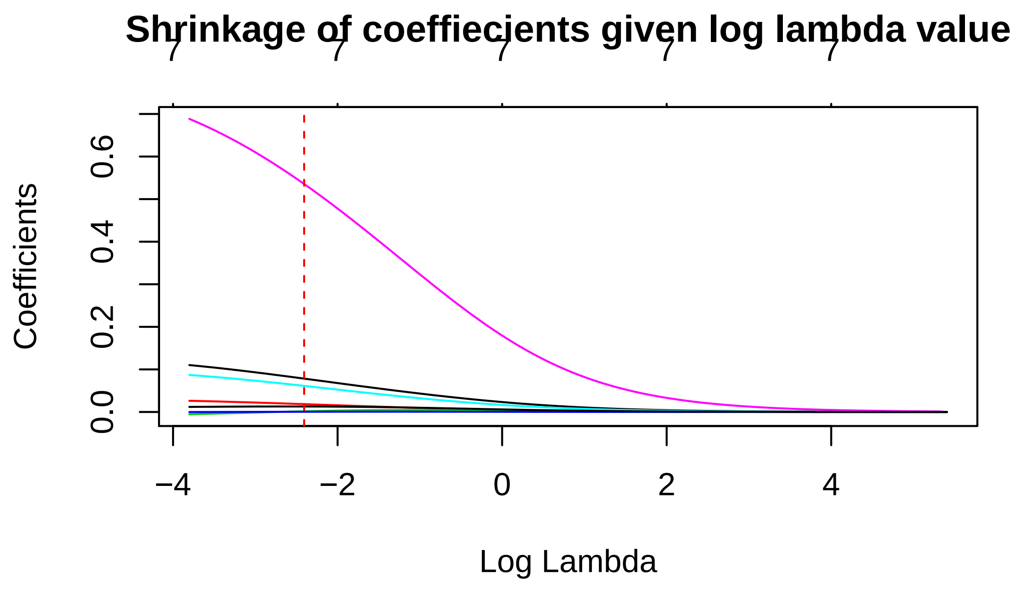
```
8 x 1 sparse Matrix of class "dgCMatrix"
                                    1
(Intercept)              -4.8297019271
Pregnancies               0.0569669262
Glucose                   0.0129887094
BloodPressure             0.0041530585
Insulin                   0.0006732565
BMI                       0.0434579592
DiabetesPedigreeFunction  0.4125272707
Age                       0.0116916136
```

Ridge regression shrinks parameters to zero, without making the estimates have that exact value. We can observe that *Insulin* related coefficient is really low in value, while the *DiabetesPedigreeFunction* coefficient has the larger magnitude. It's so useful for preventing overfitting, but not useful for feature selection. In the following plot we can observe the shrinkage of coefficients. The red line indicated the log(lambda) that produces best accuracy.

```
plot(model_ridge, main="Misclassification error
     for Ridge Regression given log lambda value")
```



```
z=glmnet(train_for_shrinkage, train$Outcome,

alpha = 0, family = "binomial", type.measure = "class")
plot(z, xvar="lambda",
     main="Shrinkage of coeffiecients given log lambda value")
abline(v = log(model_ridge$lambda.min), col = "red",lty=2)
```



We can use the model to make predictions and compute our metrics of interest:

```
pred_ridge<- predict(model_ridge,test_for_shrinkage,
                     type = "response", s = model_ridge$lambda.min)
pred_ridger<-ifelse(pred_ridge > 0.3, 1, 0)
```

```
#Confusion matrix
c_ridge=table(test$Outcome, pred_ridger)
```

Accuracy:

```
accuracy_ridge = (c_ridge[1,1]+c_ridge[2,2])/nrow(test)

accuracy_ridge
```

```
[1] 0.6703297
```

Sensitivity:

```
sensitivity_ridge = c_ridge[2,2]/(c_ridge[2,2]+c_ridge[2,1])
```

```
sensitivity_ridge
```

```
[1] 0.8245614
```

Lasso Regression

```
model_lasso=cv.glmnet(train_for_shrinkage, train$Outcome,

alpha = 1, family = "binomial", type.measure = "class")
model_lasso
```

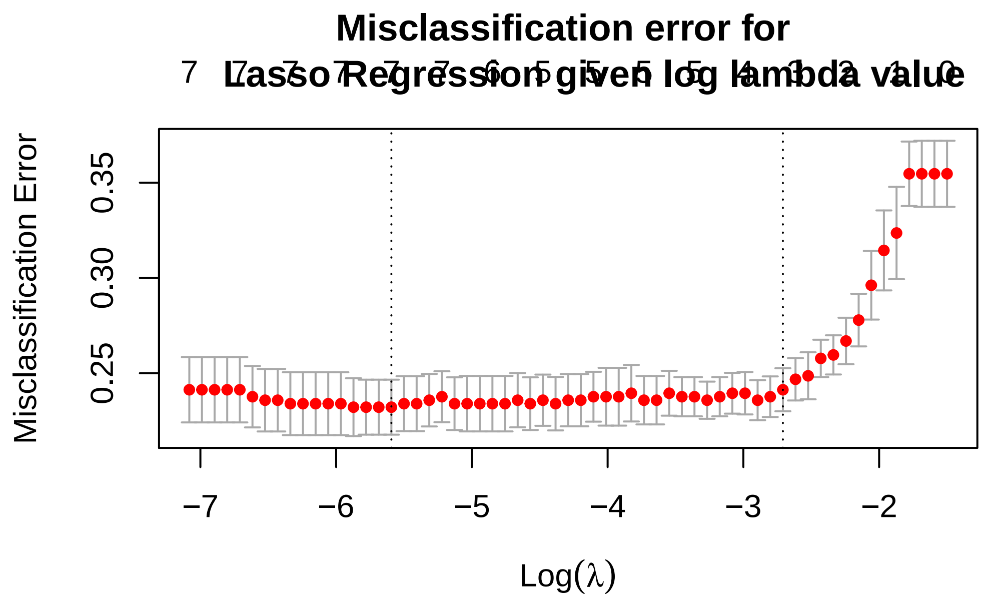
```
Call:  cv.glmnet(x = train_for_shrinkage, y = train$Outcome, type.measure = "class",      al

Measure: Misclassification Error

     Lambda Index Measure      SE Nonzero
min 0.00372    45  0.2322 0.01443       7
1se 0.06660    14  0.2413 0.01129       3
```

```
coef(model_lasso)
```

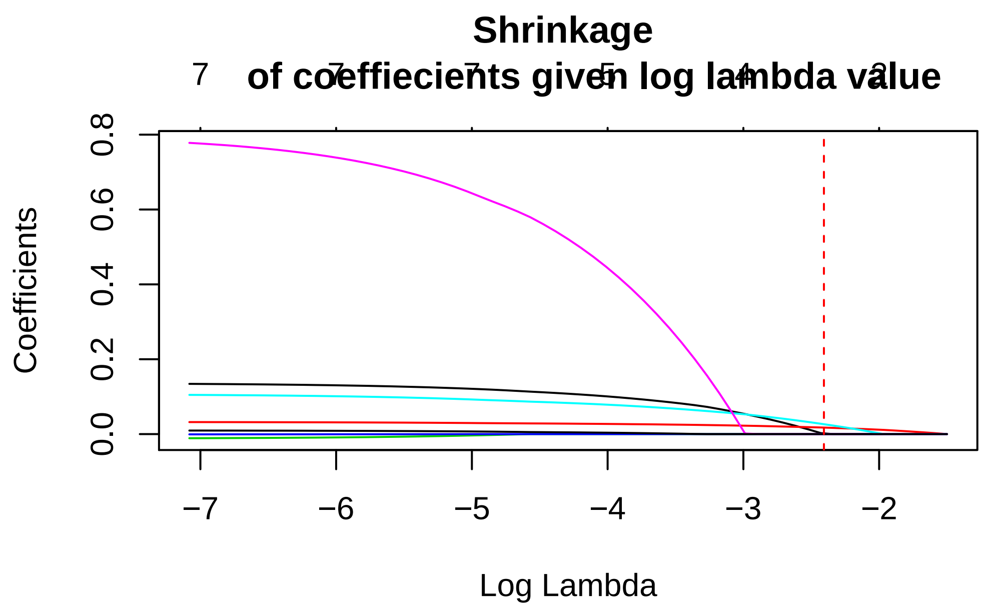
```
8 x 1 sparse Matrix of class "dgCMatrix"
                                  1
(Intercept)              -4.61136855
Pregnancies               0.03080965
Glucose                   0.02024480
BloodPressure             .
Insulin                   .
BMI                       0.04127385
DiabetesPedigreeFunction  .
Age                       .
```

As we can see, Lasso regression shrinks to zero parameter related to variables *BloodPressure*, *Insulin*, *DiabetesPedigreeFunction, Age* has a really low value, given feature's collinearity with *Pregnancies.* As we expected, Lasso yields sparse models.

```
plot(model_lasso,main="Misclassification error for
     Lasso Regression given log lambda value")
```



```
z=glmnet(train_for_shrinkage, train$Outcome,

alpha = 1, family = "binomial", type.measure = "class")
plot(z, xvar="lambda",main="Shrinkage
     of coeffiecients given log lambda value")
abline(v = log(model_ridge$lambda.min), col = "red",lty=2)
```



```
pred_lasso<- predict(model_lasso,test_for_shrinkage,
                     type = "response", s = model_lasso$lambda.min)
pred_lassor<-ifelse(pred_lasso > 0.3, 1, 0)
```

```
#Confusion matrix
c_lasso=table(test$Outcome, pred_lassor)
c_lasso
```

```
   pred_lassor
     0  1
  0 83 42
  1 15 42
```

Accuracy:

```
accuracy_lasso = (c_lasso[1,1]+c_lasso[2,2])/nrow(test)
accuracy_lasso
```

```
[1] 0.6868132
```

Sensitivity:

```
sensitivity_lasso = c_lasso[2,2]/(c_lasso[2,2]+c_lasso[2,1])
sensitivity_lasso
```

```
[1] 0.7368421
```

Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis (LDA) is a statistical technique used for classification. It assumes that the predictor variables are linearly related to the class labels. It is based on the principles of Bayesian inference and aims to find a linear combination of predictor variables that maximizes the separation between different classes. This linear combination is known as the discriminant function. Once the discriminant function is obtained, it can be used to classify new observations into the predefined classes by assigning them to the class with the highest discriminant score. The discriminant scores obtained from LDA can be interpreted as the logarithm of the posterior odds ratio.

Linear discriminant analysis is based on assumptions to ensure the correctness of the results:

- Linearly separated classes: unlike logistic regression, LDA demands that the classes can be linearly separated.
- Multivariate normality: The LDA model is more stable than the logistic regression model when the numerosity is small and the distribution of the predictors in each of the classes is approximately normal.
- Homoscedasticity: we assume that all the covariates have the same variance.
- Independence of the predictors: the LDA model works when the measurements are made on independent variables, so we have to exclude the variables with which we have the problem of collinearity. *Age* won't be used for this method.

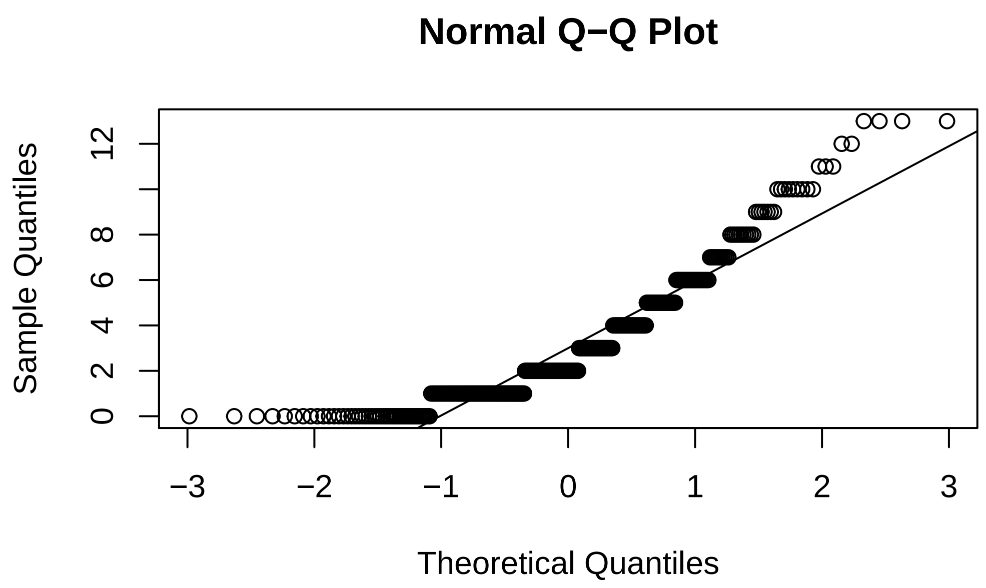
Additionally, LDA is highly sensitive to the presence of outliers, so an analysis must be done regarding these.

Normality check

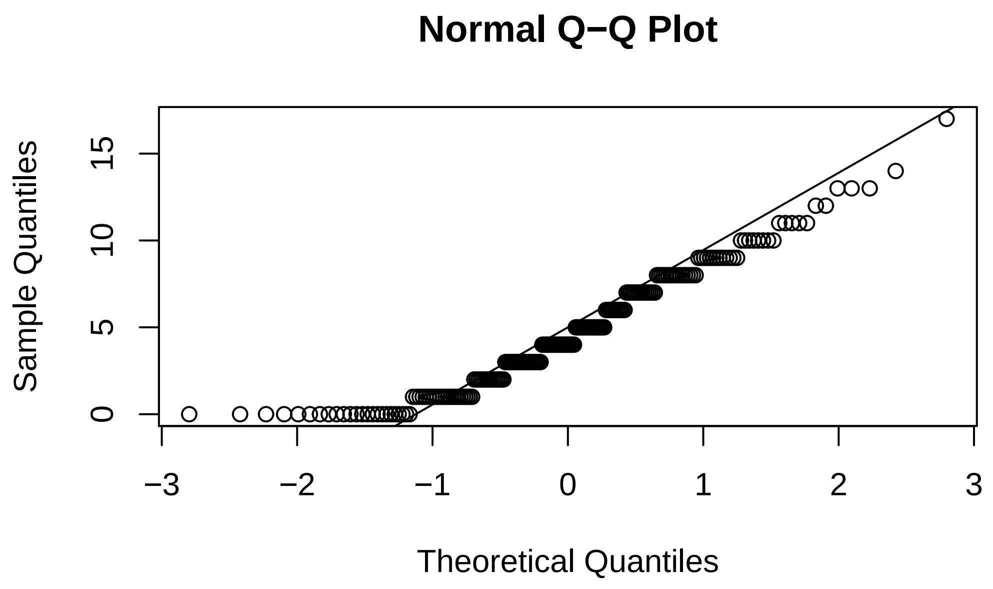
Through the use of Q-Q plot, we analyze whether the distribution of predictors in each of the "*Outcome*" classes is approximately normal

- "*Pregnancies*":

```
qqnorm(train$Pregnancies[train$Outcome==0])
qqline(train$Pregnancies[train$Outcome==0])
```



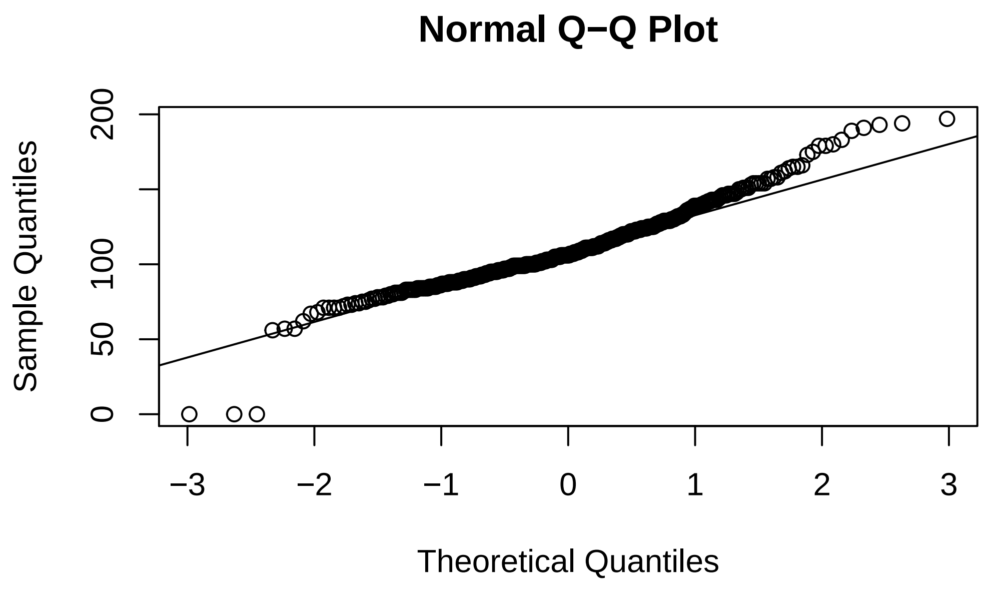
```
qqnorm(train$Pregnancies[train$Outcome==1])
qqline(train$Pregnancies[train$Outcome==1])
```



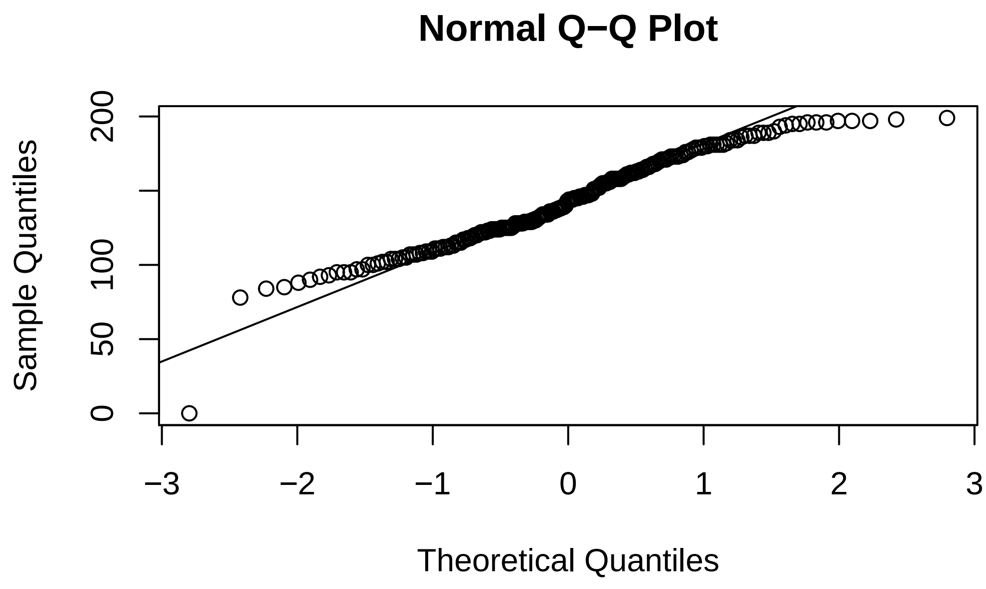
Regarding the variable "*Pregnancies*", it seems to follow approximately the Normal distribution.

- "*Glucose*":

```
qqnorm(train$Glucose[train$Outcome==0])
qqline(train$Glucose[train$Outcome==0])
```



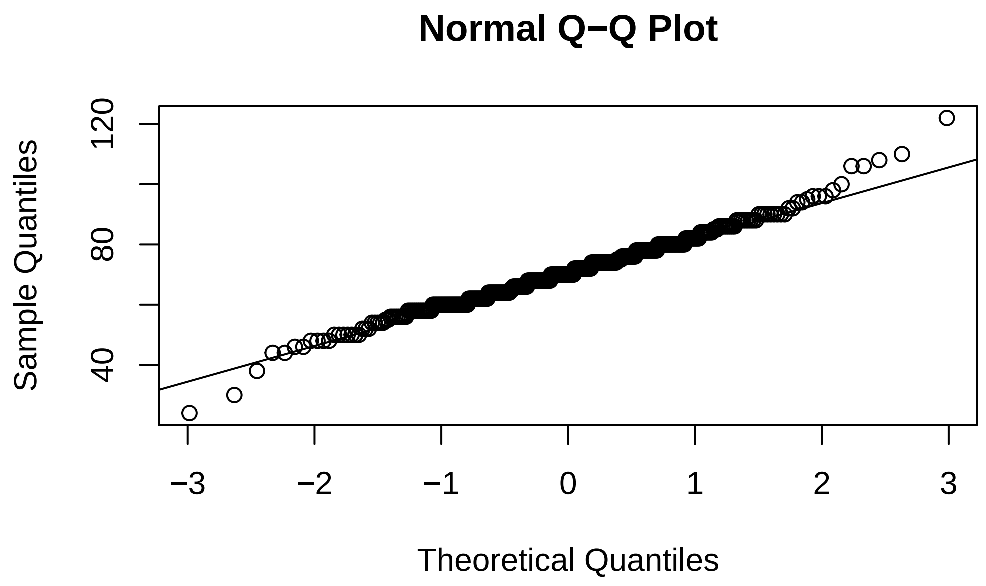
```r
qqnorm(train$Glucose[train$Outcome==1])
qqline(train$Glucose[train$Outcome==1])
```



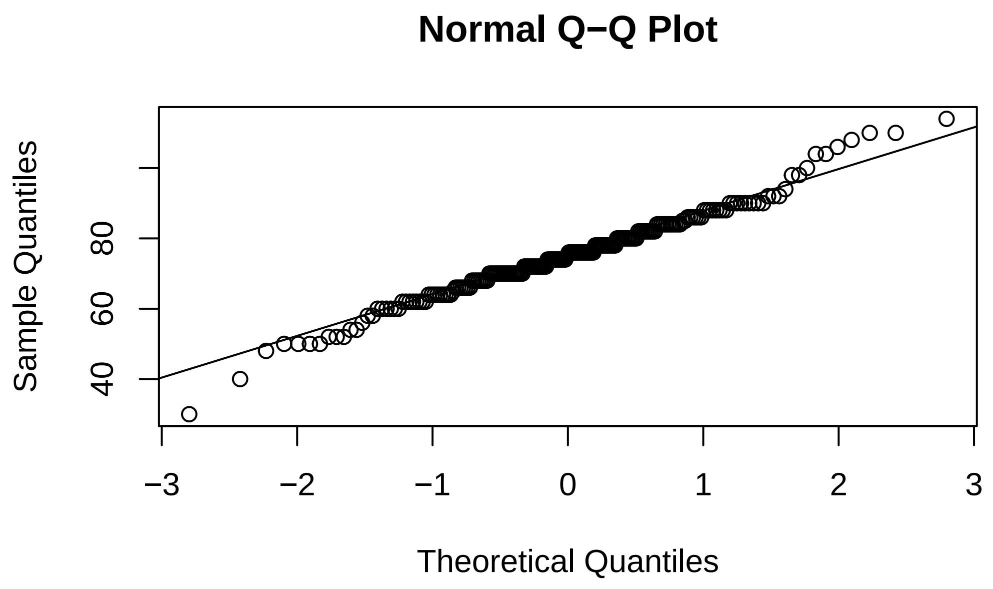
"*Glucose*"'s distribution fits very well with normal distribution.

- "*BloodPressure*":

```
qqnorm(train$BloodPressure[train$Outcome==0])
qqline(train$BloodPressure[train$Outcome==0])
```



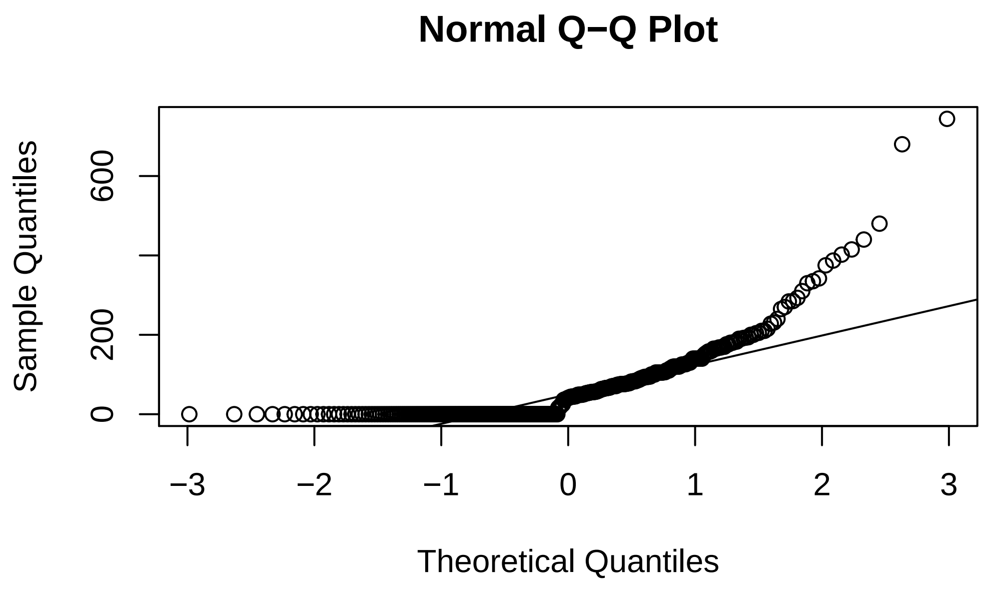
```
qqnorm(train$BloodPressure[train$Outcome==1])
qqline(train$BloodPressure[train$Outcome==1])
```



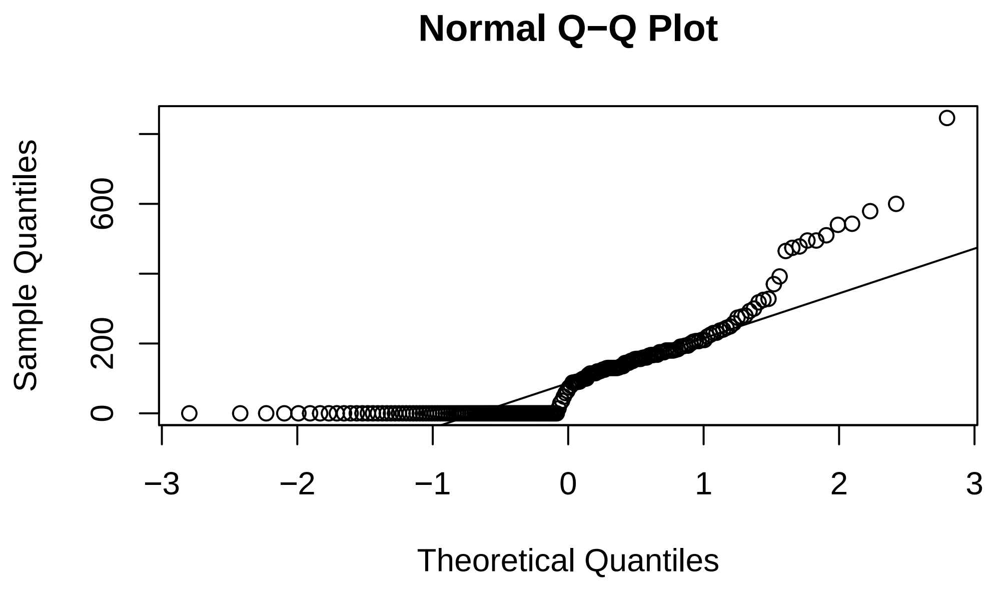
The variable "*BloodPressure*" also has a distribution approximately close to the Normal

- "*Insulin*":

```
qqnorm(train$Insulin [train$Outcome==0])
qqline(train$Insulin [train$Outcome==0])
```



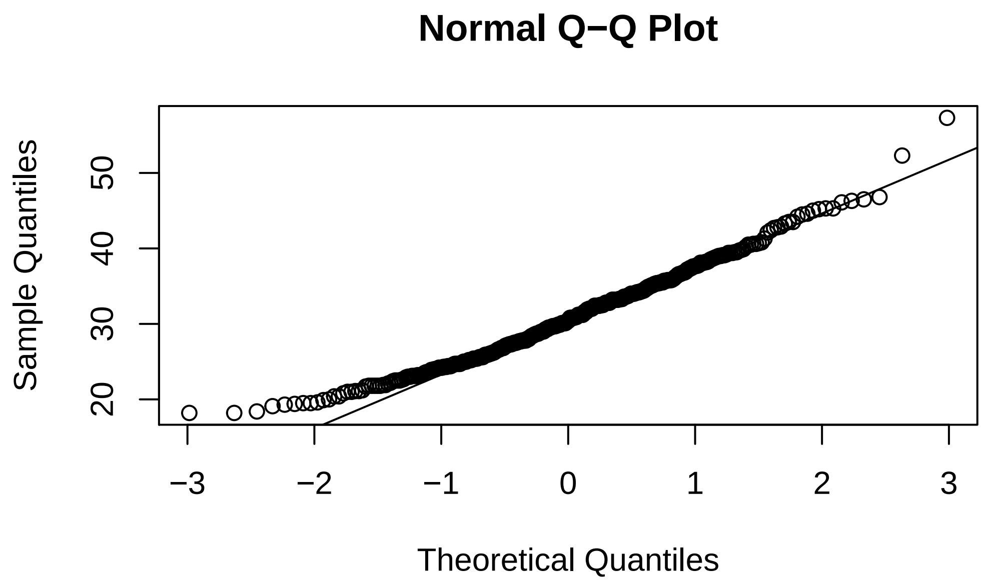
```
qqnorm(train$Insulin [train$Outcome==1])
qqline(train$Insulin [train$Outcome==1])
```



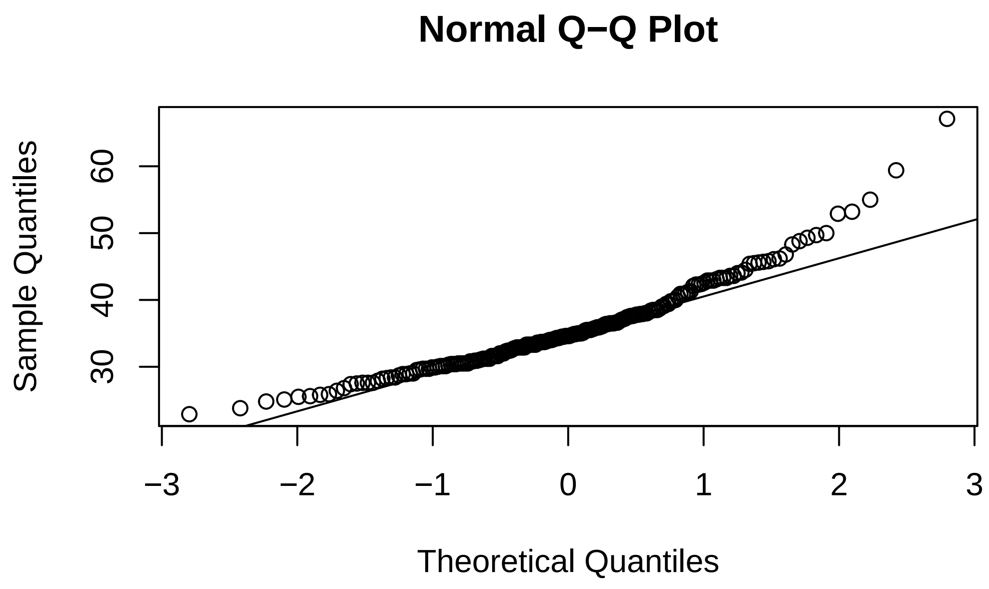
The variable "*Insulin*" does not fit the Normal distribution very well, but it seems to be the only one so far.

- "*BMI*":

```
qqnorm(train$BMI [train$Outcome==0])
qqline(train$BMI [train$Outcome==0])
```



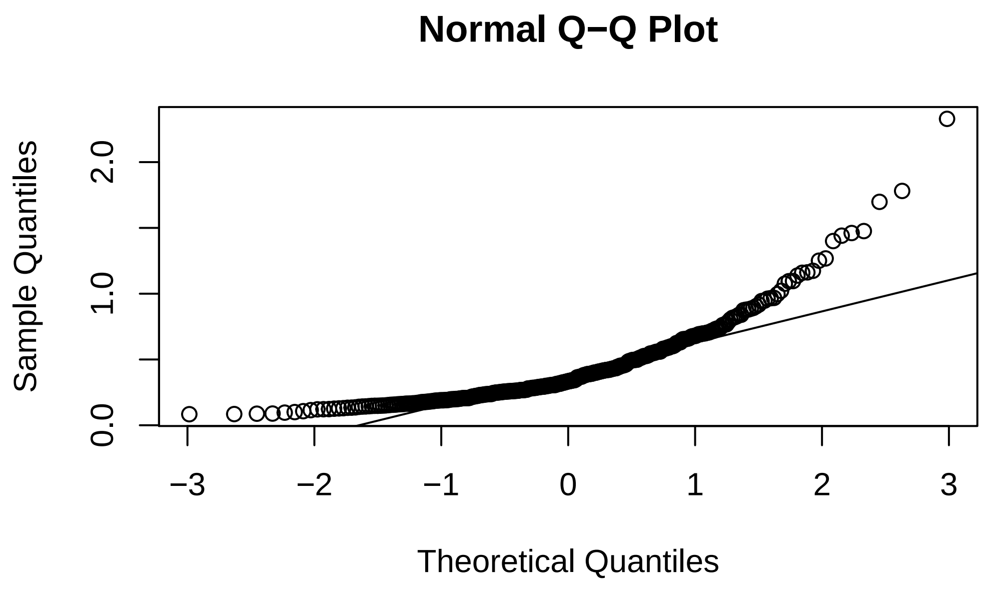
```
qqnorm(train$BMI [train$Outcome==1])
qqline(train$BMI [train$Outcome==1])
```



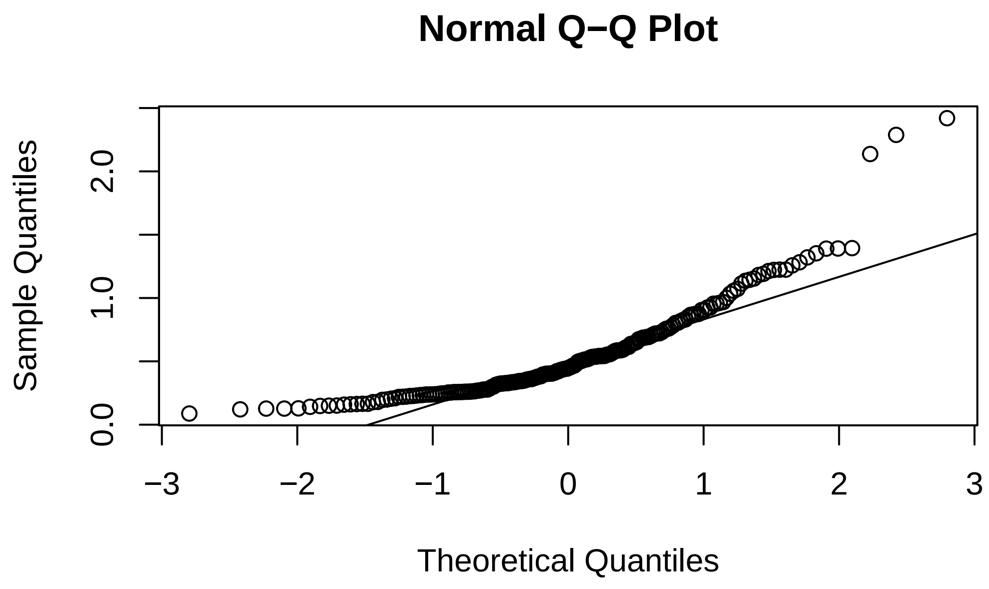
"*BMI*"'s distribution has a great fit on the Normal.

- "*DiabetesPedigreeFunction*":

```
qqnorm(train$DiabetesPedigreeFunction [train$Outcome==0])
qqline(train$DiabetesPedigreeFunction [train$Outcome==0])
```



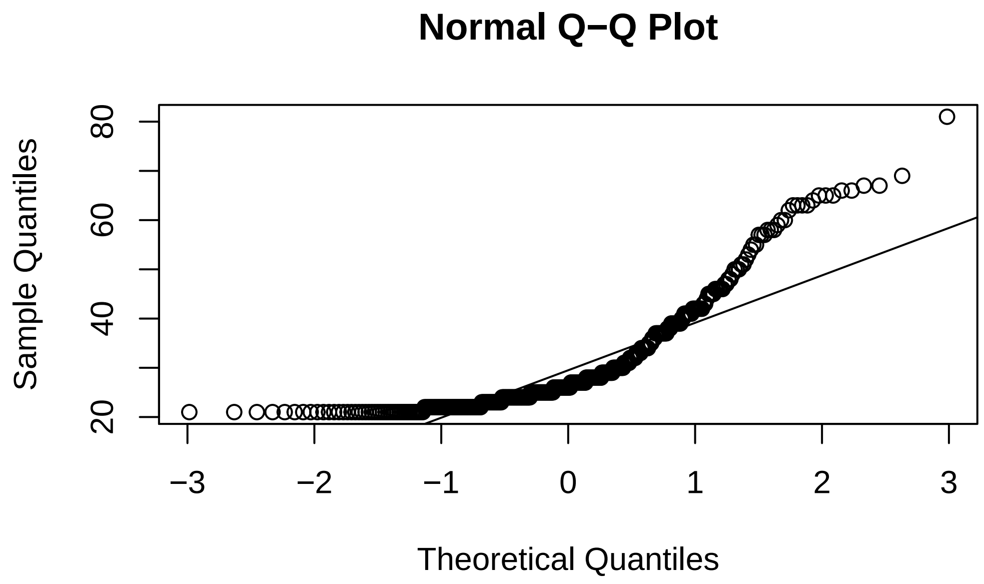
```
qqnorm(train$DiabetesPedigreeFunction [train$Outcome==1])
qqline(train$DiabetesPedigreeFunction [train$Outcome==1])
```



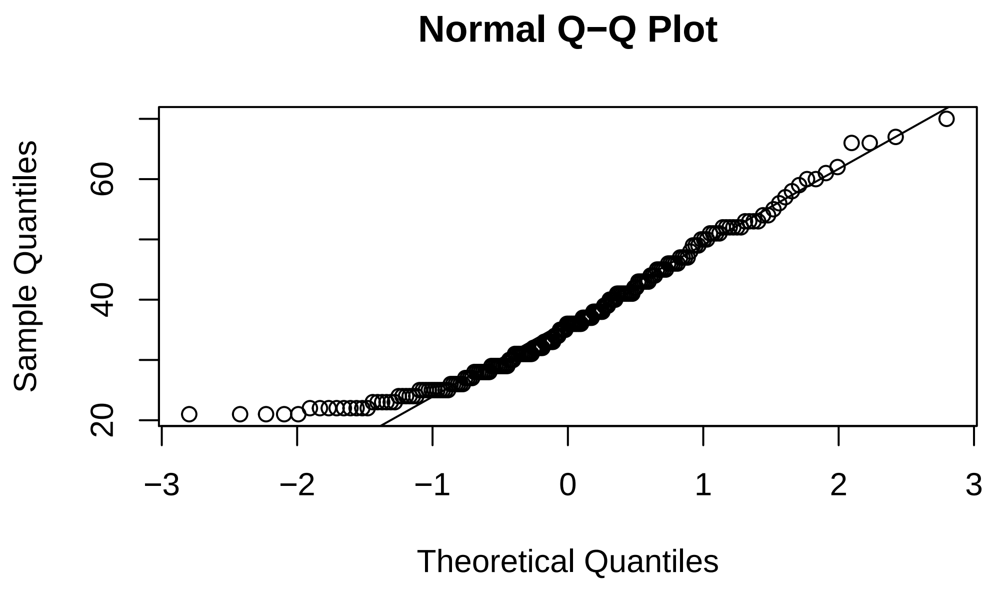
"*DiabetesPedigreeFunction*" has a Gaussian-like distribution there is the presence of some outliers that modify the tails of the distribution from the normal.

- "*Age*":

```
qqnorm(train$Age [train$Outcome==0])
qqline(train$Age [train$Outcome==0])
```



```
qqnorm(train$Age [train$Outcome==1])
qqline(train$Age [train$Outcome==1])
```



Like "*Insulin*", "*Age*" also does not fit the Gaussian curve optimally; but, overall, we can be satisfied with the multivariate normality concerning our variables.

Outliers Detection and Elimination

As reported earlier, it's needed to conduct an analysis to detect the presence of outliers, as these may affect LDA results. To do this, it's exploited the interquartile rule (IQR): the interquartile rule is based on the interquartile range: it is a measure of data dispersion that represents the difference between the third quartile (Q3) and the first quartile (Q1). The interquartile rule identifies outliers as those points that are above or below a certain range relative to the IQR.

Firstly, a new training set is defined, a copy of the original, which will be modified whenever outliers are identified:

```
#Training set without outliers
train_wo_out = train
```

Next, IQR is applied to identify any outliers contained within our predictors. Once detected, they are excluded from "*train_wo_out*".

```r
Q1_preg <- quantile(train_wo_out$Pregnancies, 0.25)
Q3_preg <- quantile(train_wo_out$Pregnancies, 0.75)

IQR_preg = Q3_preg-Q1_preg
inf_lim_preg <- Q1_preg - 1.5 * IQR_preg
sup_lim_preg <- Q3_preg + 1.5 * IQR_preg

outliers_preg <- train_wo_out$Pregnancies[train_wo_out$Pregnancies <
                                            inf_lim_preg |
                                            train_wo_out$Pregnancies > sup_lim_preg]
length(outliers_preg)
```

```
[1] 2
```

```r
train_wo_out = train_wo_out[train_wo_out$Pregnancies
                            >= inf_lim_preg &
                              train_wo_out$Pregnancies
                            <= sup_lim_preg,]
```

```r
Q1_gluc <- quantile(train_wo_out$Glucose, 0.25)
Q3_gluc <- quantile(train_wo_out$Glucose, 0.75)

IQR_gluc = Q3_gluc-Q1_gluc
inf_lim_gluc <- Q1_gluc - 1.5 * IQR_gluc
sup_lim_gluc <- Q3_gluc + 1.5 * IQR_gluc
outliers_gluc <- train_wo_out$Glucose[train_wo_out$Glucose <
                                        inf_lim_gluc |
                                        train_wo_out$Glucose > sup_lim_gluc]
length(outliers_gluc)
```

```
[1] 4
```

```r
train_wo_out = train_wo_out[train_wo_out$Glucose
                            >= inf_lim_gluc &
                              train_wo_out$Glucose
                            <= sup_lim_gluc,]
```

```
Q1_bp <- quantile(train_wo_out$BloodPressure, 0.25)
Q3_bp <- quantile(train_wo_out$BloodPressure, 0.75)

IQR_bp = Q3_bp-Q1_bp
inf_lim_bp <- Q1_bp - 1.5 * IQR_bp
sup_lim_bp <- Q3_bp + 1.5 * IQR_bp
outliers_bp <- train_wo_out$BloodPressure[train_wo_out$BloodPressure <
                                            inf_lim_bp |
                                            train_wo_out$BloodPressure > sup_lim_bp]
length(outliers_bp)
```

```
[1] 14
```

```
train_wo_out = train_wo_out[train_wo_out$BloodPressure
                            >= inf_lim_bp &
                              train_wo_out$BloodPressure
                            <= sup_lim_bp,]
```

```
Q1_ins <- quantile(train_wo_out$Insulin, 0.25)
Q3_ins <- quantile(train_wo_out$Insulin, 0.75)

IQR_ins = Q3_ins-Q1_ins
inf_lim_ins <- Q1_ins - 1.5 * IQR_ins
sup_lim_ins <- Q3_ins + 1.5 * IQR_ins
outliers_ins <- train_wo_out$Insulin[train_wo_out$Insulin <
                                       inf_lim_ins |
                                       train_wo_out$Insulin >
                                       sup_lim_ins]
length(outliers_ins)
```

```
[1] 26
```

```
train_wo_out = train_wo_out[train_wo_out$Insulin
                            >= inf_lim_ins
                            & train_wo_out$Insulin
                            <= sup_lim_ins,]
```

```
Q1_BMI <- quantile(train_wo_out$BMI, 0.25)
Q3_BMI <- quantile(train_wo_out$BMI, 0.75)

IQR_BMI = Q3_BMI-Q1_BMI
inf_lim_BMI <- Q1_BMI - 1.5 * IQR_BMI
sup_lim_BMI <- Q3_BMI + 1.5 * IQR_BMI
outliers_BMI <- train_wo_out$BMI[train_wo_out$BMI < inf_lim_BMI |
                                   train_wo_out$BMI >
                                   sup_lim_BMI]
length(outliers_BMI)
```

```
[1] 6
```

```
train_wo_out = train_wo_out[train_wo_out$BMI
                            >= inf_lim_BMI &
                              train_wo_out$BMI
                            <= sup_lim_BMI,]
```

```
Q1_dpf <- quantile(train_wo_out$DiabetesPedigreeFunction, 0.25)
Q3_dpf <- quantile(train_wo_out$DiabetesPedigreeFunction, 0.75)

IQR_dpf = Q3_dpf-Q1_dpf
inf_lim_dpf <- Q1_dpf - 1.5 * IQR_dpf
sup_lim_dpf <- Q3_dpf + 1.5 * IQR_dpf
outliers_dpf <- train_wo_out$DiabetesPedigreeFunction[train_wo_out$
                                  DiabetesPedigreeFunction
                                  < inf_lim_dpf |
                                  > sup_lim_dpf]
length(outliers_dpf)
```

```
[1] 29
```

```
train_wo_out = train_wo_out[train_wo_out$DiabetesPedigreeFunction
                            >= inf_lim_dpf &
                              train_wo_out$DiabetesPedigreeFunction
                            <= sup_lim_dpf,]
```

```r
Q1_Age <- quantile(train_wo_out$Age, 0.25)
Q3_Age <- quantile(train_wo_out$Age, 0.75)

IQR_Age = Q3_Age-Q1_Age
inf_lim_Age <- Q1_Age - 1.5 * IQR_Age
sup_lim_Age <- Q3_Age + 1.5 * IQR_Age
outliers_Age <- train_wo_out$Age[train_wo_out$Age < inf_lim_Age
                                 | train_wo_out$Age > sup_lim_Age]
length(outliers_Age)
```

```
[1] 19
```

```r
train_wo_out = train_wo_out[train_wo_out$Age
                            >= inf_lim_Age &
                              train_wo_out$Age
                            <= sup_lim_Age,]
```

```r
nrow(train)
```

```
[1] 547
```

```r
nrow(train_wo_out)
```

```
[1] 447
```

"*train_wo_out*" turns out to be less than the initial training set of 100 observations after the outliers are excluded.

LDA Procedure

To perform Linear Discriminant Analysis, we use the `lda` function built into the `MASS` package:

```r
library(MASS)
```

LDA model is defined applying the best subset of variables deriving from the analysis conduced with the use of VIF, so it doesn't contain "*Age*"; it is trained on "*train_wo_out*":

```r
lda_model <- lda(train_wo_out$Outcome~ BloodPressure+BMI+
                   Glucose+Insulin+DiabetesPedigreeFunction
                 +Pregnancies ,data=train_wo_out, family="binomial")
lda_model
```

```
Call:
lda(train_wo_out$Outcome ~ BloodPressure + BMI + Glucose + Insulin +
    DiabetesPedigreeFunction + Pregnancies, data = train_wo_out,
    family = "binomial")

Prior probabilities of groups:
        0         1
0.6800895 0.3199105

Group means:
  BloodPressure      BMI  Glucose  Insulin DiabetesPedigreeFunction Pregnancies
0      70.46711 30.71612 108.4408 56.46382                0.3918125    3.108553
1      74.21678 34.64196 140.3846 77.30769                0.4689021    4.853147

Coefficients of linear discriminants:
                                  LD1
BloodPressure            -0.005937433
BMI                       0.062925192
Glucose                   0.032153287
Insulin                  -0.001353290
DiabetesPedigreeFunction  1.094202832
Pregnancies               0.121003859
```

In the model, the linear discriminant's coefficient LD1 that is found to have greater weight in discriminating the observations between the two class of "*Outcome*" is the one related to "*DiabetesPedigreeFunction*"; an unit increase in this variable corresponds to an increase of about 1.09 in the value of the linear discriminant, advancing classification in group 1. The LD1 of "*Insulin*" and "*BloodPressure*" are the only coefficient of linear discriminants where, a rise in it, corresponds to a decrease in discriminant in the discriminant function value, favoring grouping under 0. However, both magnitudes don't make them significantly important in the classification process.

Predictions are calculated through the "predict" command. Then, we consider the posterior probabilities; these are calculated for new observations to determine which class group they are most likely to belong to. The posterior probabilities indicate the model's "confidence" in classifying an observation into a particular group.

```
# Computing predictions:
pred_lda_model<- predict(lda_model, test, type = "response")
post_lda_model<- pred_lda_model$posterior

# Converting the predictions  according to the threshold of 0.3:
pred_lda_modelr <- ifelse(post_lda_model[,2] > 0.3, 1, 0)
```

Finally, we calculate the metrics:

```
# Confusion matrix with threshold = 0.3
c_lda = table(test$Outcome,pred_lda_modelr)
c_lda
```

```
   pred_lda_modelr
     0  1
  0 88 37
  1 14 43
```

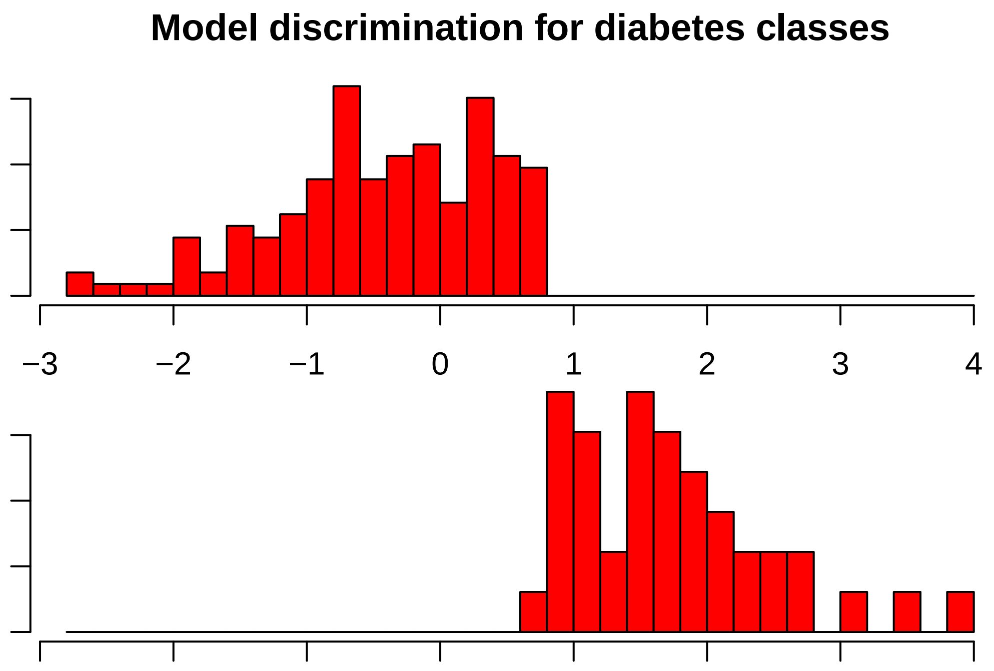
```
# Accuracy:
accuracy_lda = (c_lda[1,1]+c_lda[2,2])/nrow(test)

accuracy_lda
```

```
[1] 0.7197802
```

```
# Sensitivity:
sensitivity_lda = c_lda[2,2]/(c_lda[2,2]+c_lda[2,1])

sensitivity_lda
```

```
[1] 0.754386
```

```
par(mar=c(1, 1, 1, 1))
ldahist(pred_lda_model$x[,1], pred_lda_model$class, col=2)
title("Model discrimination for diabetes classes")
```



Quadratic Discriminant Analysis

The homoscedasticity assumption of LDA is a strong assumption; it is possible to get rid of it through the Quadratic Discriminant Analysis (QDA); this technique is an extension of LDA, that estimates a mean and a variance matrix for each class separately. Since QDA calculates a variance matrix for each class, it is particularly affected by outliers and needs a larger training set; in our analysis, it is trained on "*train_wo_out*", which allows the resolution of the problem related to outliers, but turns out to have a not particularly large sample size.

To perform Quadratic Discriminant Analysis, we use the `qda` function built into the `MASS` package:

```
qda_model <- qda(train_wo_out$Outcome~BloodPressure+BMI+
                   Glucose+Insulin+DiabetesPedigreeFunction+
                   Pregnancies ,data=train_wo_out, family="binomial")

qda_model
```

```
Call:
qda(train_wo_out$Outcome ~ BloodPressure + BMI + Glucose + Insulin +
    DiabetesPedigreeFunction + Pregnancies, data = train_wo_out,
    family = "binomial")
```

```
Prior probabilities of groups:
        0         1 
0.6800895 0.3199105 

Group means:
  BloodPressure      BMI  Glucose  Insulin DiabetesPedigreeFunction Pregnancies
0      70.46711 30.71612 108.4408 56.46382                0.3918125    3.108553
1      74.21678 34.64196 140.3846 77.30769                0.4689021    4.853147
```

```
# Computing predictions:
pred_qda_model<- predict(qda_model, test, type = "response")
post_qda_model<- pred_qda_model$posterior

# Converting the predictions  according to the threshold of 0.3:
pred_qda_modelr <- ifelse(post_qda_model[,2] > 0.3, 1, 0)
```

```
# Confusion matrix with threshold = 0.3
c_qda = table(test$Outcome,pred_qda_modelr)
c_qda
```

```
   pred_qda_modelr
     0  1
  0 80 45
  1 17 40
```

```
# Accuracy:
accuracy_qda = (c_qda[1,1]+c_qda[2,2])/nrow(test)

accuracy_qda
```

```
[1] 0.6593407
```

```
# Sensitivity:
sensitivity_qda = c_qda[2,2]/(c_qda[2,2]+c_qda[2,1])

sensitivity_qda
```

```
[1] 0.7017544
```

The metrics computed for QDA result are worse than LDA metrics; both for accuracy and sensitivity. The reason behind this could be that QDA requires higher quantity of data.

K-NN

K-Nearest Neighbors is a non-parametric classification method that classify an individual to diabetes class or not given the majority class among k-nearest points in the feature. It's important to choose a k that maximizes a chosen metric. Given the unbalanceness of the data, we try k values until 100: an higher k would underfit the data, labeling all the examples as the majority class.

```
accuracy_all_knn=vector()
sensitivity_all_knn=vector()
for (k in 1:100) {
    knn_=knn(train[,1:7], test[,1:7], cl = train$Outcome, k = k)
    c=table(knn_,test$Outcome)
    accuracy_iter=(c[1,1]+c[2,2])/nrow(test)
    sensitivity_iter=c[2,2]/(c[2,2]+c[2,1])
    accuracy_all_knn=c(accuracy_all_knn,accuracy_iter)
    sensitivity_all_knn=c(sensitivity_all_knn,sensitivity_iter)

    }
print("ACCURACY FOR BEST K:")
```

```
[1] "ACCURACY FOR BEST K:"
```

```
print(max(accuracy_all_knn))
```

```
[1] 0.7912088
```

```
print("K=")
```

```
[1] "K="
```

```
print(which.max(accuracy_all_knn))
```

```
[1] 28
```

```r
print("###########################")
```

```
[1] "###########################"
```

```r
print("SENSITIVITY FOR BEST K:")
```

```
[1] "SENSITIVITY FOR BEST K:"
```

```r
print(max(sensitivity_all_knn))
```

```
[1] 0.7209302
```

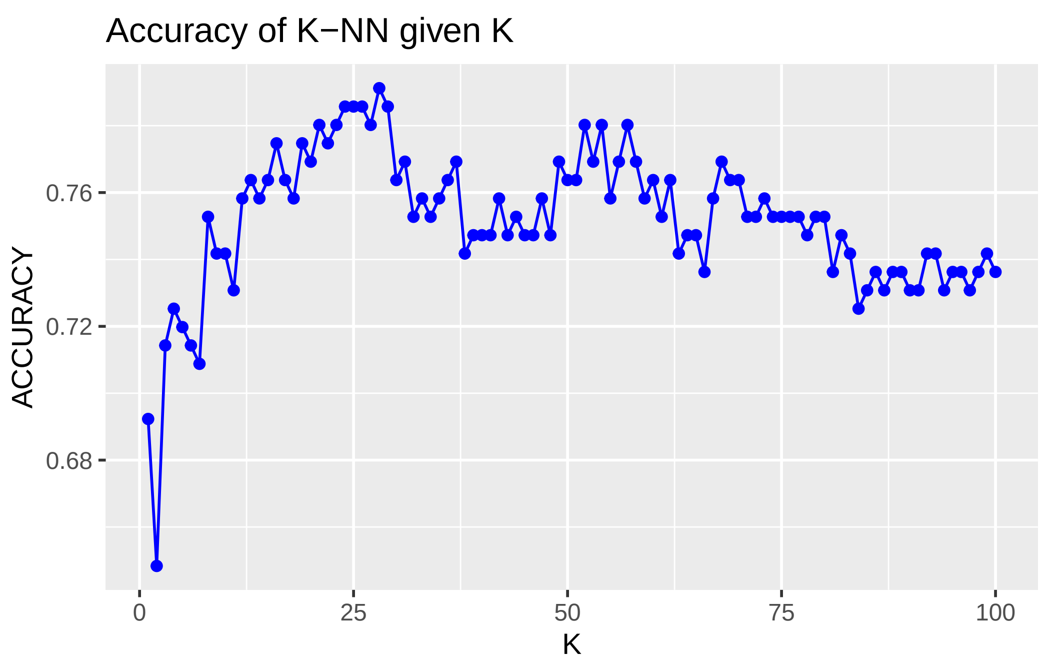
```r
print("K=")
```

```
[1] "K="
```

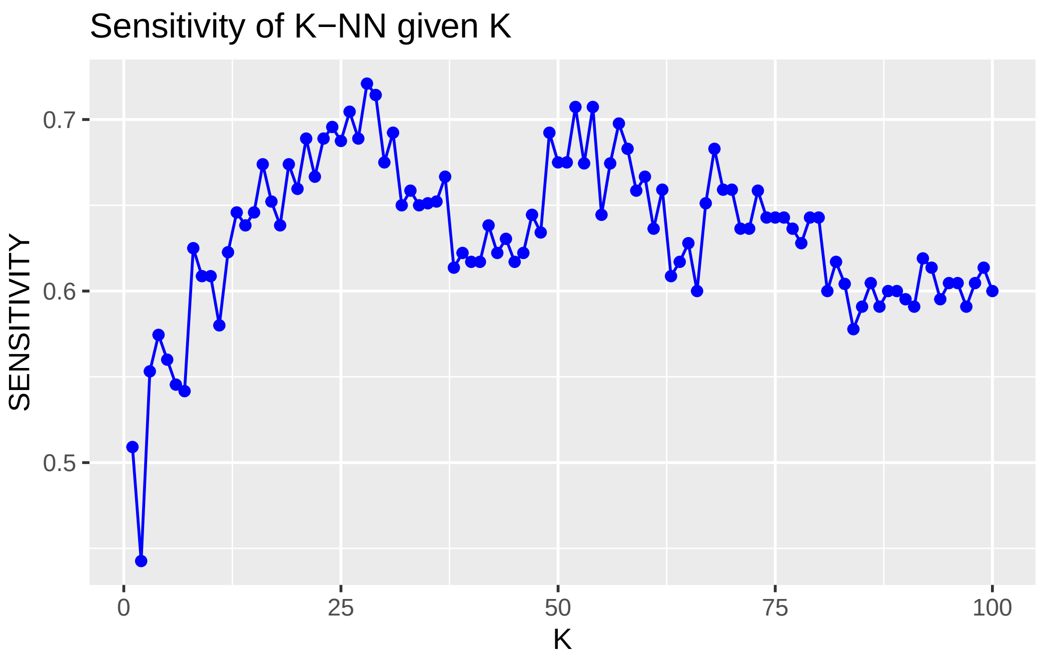
```r
print(which.max(sensitivity_all_knn))
```

```
[1] 28
```

```r
accuracy_k=data.frame(K=1:100,ACCURACY=accuracy_all_knn)
ggplot(accuracy_k, aes(x = K, y = ACCURACY)) +
geom_line(colour="blue") +
geom_point(colour="blue") + ggtitle("Accuracy of K-NN given K")
```



```
sensitivity_k=data.frame(K=1:100, SENSITIVITY=sensitivity_all_knn)
ggplot(sensitivity_k, aes(x = K, y = SENSITIVITY)) +
geom_line(colour="blue") +
geom_point(colour="blue") + ggtitle("Sensitivity of K-NN given K")
```



We can observe a really nice result, since k=28 produces both best accuracy and best sensitivity.

Model Comparison

ROC Curve

It's introduced the ROC (Receiver Operating Characteristic) curve, a tool used to evaluate and visualize the performance of a classification model. The ROC curve represents on the y-axis the relationship between the sensitivity (True Positive Rate) and on the x-axis, 1 - the specificity (True Negative Rate) of the model. In order to choose the best model, it is considered the one that maximizes the area subtended by the curve (AUC). The AUC ranges from 0 to 1: a value closer to 1 denotes better predictive ability of the model; an AUC of 0.5 indicates that the model cannot discriminate better than random choice. In our scenario, we give greater importance to those models that have high Sensitivity values, in addition to high AUC values.

```r
roc_logistic_b = roc(test$Outcome,
                      as.numeric(pred_model_back), levels=c(0, 1))
```

```
Setting direction: controls < cases
```

```r
plot(roc_logistic_b, legacy.axes=TRUE,
     main = "Curva ROC", print.auc = FALSE, xlim=c(1.2, -0.2))


roc_logistic_c = roc(test$Outcome,
                     as.numeric(pred_model_1), levels=c(0, 1))
```

```
Setting direction: controls < cases
```

```r
lines(roc_logistic_c, col = "purple", lty = 1)


roc_logistic_f = roc(test$Outcome, pred_model_for, levels=c(0, 1))
```

```
Setting direction: controls < cases
```

```r
lines(roc_logistic_f, col = "red", lty = 1)


roc_lda = roc(test$Outcome,
              as.numeric(post_lda_model[,2]), levels=c(0, 1))
```

```
Setting direction: controls < cases
```

```r
lines(roc_lda, col = "green", lty = 1)


roc_qda = roc(test$Outcome,
              as.numeric(post_qda_model[,2]), levels=c(0, 1))
```

```
Setting direction: controls < cases
```

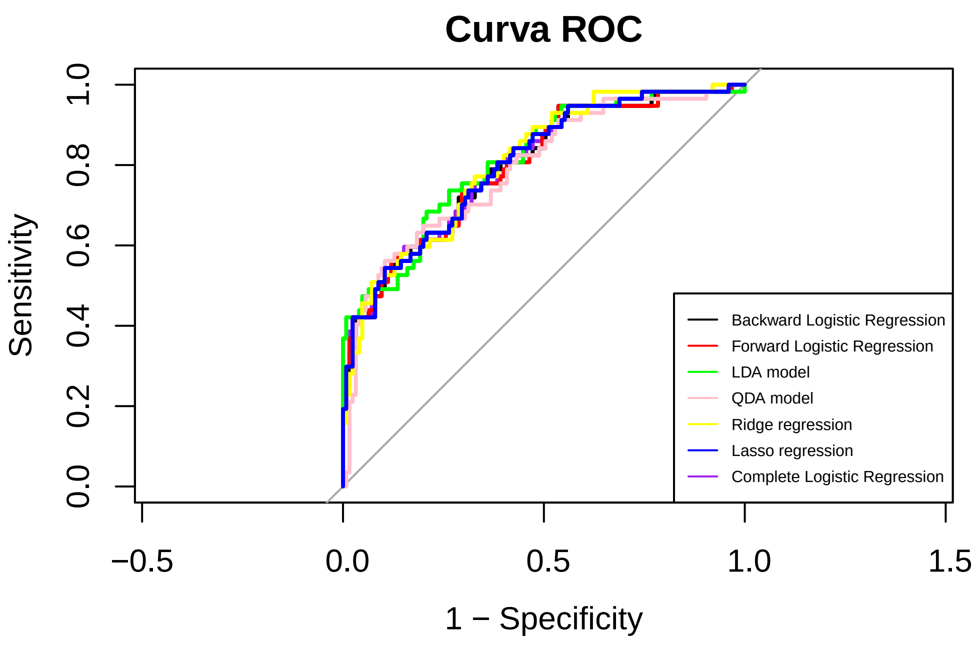
```r
lines(roc_qda, col = "pink", lty = 1)


roc_ridge = roc(test$Outcome,
                  as.numeric(pred_ridge), levels=c(0, 1))
```

```
Setting direction: controls < cases
```

```r
lines(roc_ridge, col = "yellow", lty = 1)


roc_lasso = roc(test$Outcome,
                  as.numeric(pred_lasso), levels=c(0, 1))
```

```
Setting direction: controls < cases
```

```r
lines(roc_lasso, col = "blue", lty = 1)

legend("bottomright", legend = c("Backward Logistic Regression",
                                 "Forward Logistic Regression",
                                 "LDA model", "QDA model",
                                 "Ridge regression",
                                 "Lasso regression",
                                 "Complete Logistic Regression"),
       col = c("black", "red", "green", "pink",
               "yellow",  "blue", "purple"), lty = 1, cex = 0.5)
```



```r
print("AUC LOGISTIC WITH BACK ELIMINATION")
```

```
[1] "AUC LOGISTIC WITH BACK ELIMINATION"
```

```r
auc(test$Outcome,pred_model_back)[1]
```

```
Setting levels: control = 0, case = 1

Setting direction: controls < cases

[1] 0.7977544
```

```r
print("AUC LOGISTIC WITH ALL FEATURES")
```

```
[1] "AUC LOGISTIC WITH ALL FEATURES"
```

```r
auc(test$Outcome,pred_model_1)[1]
```

```
Setting levels: control = 0, case = 1

Setting direction: controls < cases

[1] 0.802386
```

```
print("AUC LOGISTIC WITH FORWARD SELECTION")
```

```
[1] "AUC LOGISTIC WITH FORWARD SELECTION"
```

```
auc(test$Outcome,pred_model_for)[1]
```

```
Setting levels: control = 0, case = 1

Setting direction: controls < cases

[1] 0.7977544
```

```
print("AUC LDA")
```

```
[1] "AUC LDA"
```

```
auc(test$Outcome,post_lda_model[,2])[1]
```

```
Setting levels: control = 0, case = 1

Setting direction: controls < cases

[1] 0.8122105
```

```
print("AUC QDA")
```

```
[1] "AUC QDA"
```

```r
auc(test$Outcome,post_qda_model[,2])[1]
```

```
Setting levels: control = 0, case = 1

Setting direction: controls < cases

[1] 0.7904561
```

```r
print("AUC RIDGE REGRESSION")
```

```
[1] "AUC RIDGE REGRESSION"
```

```r
auc(test$Outcome,as.numeric(pred_ridge))[1]
```

```
Setting levels: control = 0, case = 1

Setting direction: controls < cases

[1] 0.8064561
```

```r
print("AUC LASSO REGRESSION")
```

```
[1] "AUC LASSO REGRESSION"
```

```r
auc(test$Outcome,as.numeric(pred_lasso))[1]
```

```
Setting levels: control = 0, case = 1

Setting direction: controls < cases

[1] 0.8019649
```

We can observe how all models have overall the same AUC, with the LDA that is marginally better.

Metrics Comparison

The model that achieve the best AUC is the LDA model, but as we can see the AUC values are really closer to each other. Moreover, given the purpose of our analysis, we would prefer a model with the higher True Positive Rate (Sensitivity) without caring much about Sensibility (True Negative Rate).

The following table resumes the accuracy and sensitivity reached by each model we estimated:

Model	Accuracy	Sensitivity
Complete Logistic Model	0.6923	0.7368
Backward Logistic Model	0.6923	0.7368
Forward Logistic Model	0.7143	0.7544
Ridge Regression Model	0.6703	0.8246
Lasso Regression Model	0.6868	0.7368
LDA Model	0.7198	0.7544
QDA Model	0.6593	0.7018
28-NN	0.7912	0.7209

We can group the models into 5 groups:

1. Complete Logistic Model, Backward Logistic Model and Lasso Regression: these models achieve similar and balanced values for both metrics.
2. Forward Logistic and LDA, which represent a clear update from the previous group: even in this case we have balanced metrics but with an overall improvement.
3. 28-NN achieves the best Accuracy, but in terms of Sensitivity it gets worse than the previous two groups.
4. QDA is useless for our aim: QDA achieves worst metrics values both for Sensitivity and Accuracy.
5. The one that gives us the best Sensitivity is Ridge Regression: with about 82.5% is by far the best model in order to predict the illness. The accuracy is quite low, but for our aim is absolutely the model we would use in practice.

Conclusions

Basically all the model estimated suggest how the presence of diabetes is highly correlated with four main factors:

- The genetics susceptibility, expressed by *DiabetesPedigreeFunction*. This result is largely confirmed by medical literature.
- The number of pregnancies, which taking into account the social-cultural features of our population, can be seen as a proxy for the age. In Pima Indian American population we generally have a really high birth rate. By the way the data don't enable us to know if the measurements were taken during the pregnancy of the patient. This would have been a really meaningful particular given a specific type of diabetes called gestational diabetes.
- The Body Mass Index and the Glucose level in the bloodstream. Given the nature of the illness, overweight females clearly have higher chances to be affected by diabetes. Moreover the glucose level is directly linked with the diabetes: when the value is over 200, the diagnosis is certain.

It's important to note that the higher magnitude of coefficients related to *DiabetesPedigree-Function*, in basically every estimated model, is caused by its limited range compared to the other three features. An unit increase in the diabetes pedigree function covers slightly less than half its whole range, while in *BMI* and *Pregnancies* and especially in *Glucose* an unit increase is really relative.